UNIVERSIDADE D
COIMBRA

Guilherme Pedro Antunes Carvalhas Gabriel Coelho

Computational Discovery of Drug-target Interaction

Thesis submitted to the Faculty of Science and Technology of the University of Coimbra for the degree of Master in Biomedical Engineering with specialization in Clinical Informatics and Bioinformatics, supervised by Prof. Dr. Bernardete Ribeiro and Prof. Dr. Joel Arrais.

September 2018

• U  C •

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Guilherme Pedro Antunes Carvalhas Gabriel Coelho

Computational Discovery of Drug Target Interaction

Thesis submitted to the
University of Coimbra for the degree of
Master in Biomedical Engineering

Supervisors:
Prof. Dr. Joel P. Arrais (University of Coimbra)
Prof. Dra. Bernardete Ribeiro (University of Coimbra)

Coimbra, 2018

This research has been funded by the Portuguese Research Agency FCT, through D4 - Deep Drug Discovery and Deployment (CENTRO-01-0145-FEDER-029266)

"*Everything should be made simple as possible, but not simpler.*"

Albert Einstein

Resumo

A descoberta de antibióticos foi rapidamente seguida por um aumento da resistência bacterial aos mesmos. Esta resistência torna necessária a descoberta de novos fármacos, cujo processo requer tempo e esforços financeiros. O reposicionamento de fármacos foi proposto como a abordagem mais adequada para contornar esta dependência e permitir o desenvolvimento de novos fármacos. O maior desafio para o reposicionamento de fármacos prende-se com a identificação da possível interação entre fármacos conhecidos e os seus alvos. Conseguir prever a interação entre um fármaco e uma determinada entidade biológica, através de métodos computacionais, permite reduzir tanto o tempo necessário como os custos atualmente gastos em investigação farmacológica e testes clínicos.

O sucesso das abordagens tradicionais para a previsão da interação entre um fármaco e o seu alvo depende excessivamente das variáveis usadas para descrever os dados. Contudo não existe um consenso que determine quais as variáveis que têm maior impacto aquando da previsão. Assim sendo, os métodos tradicionais de *machine learning* tornam-se numa abordagem ineficaz. Neste trabalho, de forma a conseguir prever corretamente esta interação, desenvolveu-se um modelo baseado numa arquitetura de *deep learning*, dada a sua capacidade de priorizar, durante o treino, as variáveis com maior capacidade de classificação. De forma a construir um modelo eficaz os vários parâmetros da arquitetura foram ajustados.

A arquitetura proposta atingiu uma *accuracy* de 0.90, demonstrando melhor resultados que outros trabalhos na área, baseados em modelos tradicionais. Estes resultados sugerem que o modelo construído, poderá ser usado posteriormente tanto para prever a interação entre um novo fármaco e um determinado alvo biológico, como entre um fármaco existente e um novo alvo. De facto, o modelo poderá ser inserido no processo de reposicionamento de fármacos, indicando quais os melhores candidatos, o que resultaria numa redução tanto dos custos como do tempo que requer este processo.

Abstract

The discovery of antibiotics was quickly followed by the emergence of bacterial antibiotic resistance. This resistance, makes the discovery of new drugs an urgent need. *De novo* drug discovery process is an expensive and time consuming task. Drug repositioning has been proposed as the best approach for this issue. Identifying interaction between known drugs and targets is a major challenge on drug repositioning. *In silico* prediction of drug target interactions has significant potential to bring down the time and cost of the awfully expensive drug discovery research and clinical development.

Traditionally, the performance of drug target interaction prediction models depends heavily on the descriptors used, and there is no widely agreement on which drug and target descriptors have the best predictive power. This makes the use of traditional machine learning algorithms a rudimentary approach. In this work, to accurately predict new drug target interactions, we developed a deep learning based architecture, capable of understanding, during training, the best descriptors for the classification. Different parameters inside the architecture were tuned so we could construct the best model.

The proposed model reaches an accuracy of 0.90, outperforming current state-of-the-art methods based on shallow architectures. The results obtained suggest that the model could be further used to predict whether, the interaction between a new drug and an existing target, or between a new target and some existing drug. Actually, the model may be used on the identification of new leads for drug repositioning and so significantly improve the actual drug discovery process.

Keywords: deep learning, drug target interaction prediction, drug discovery, artificial neural network

List of Acronyms

ACC accuracy.
AI artificial intelligence.
ANN artificial neural network.
API Application Programming Interface.
AUC area under ROC curve.

CADD computer aided drug design.

DL deep learning.
DNN deep neural network.
DT decision tree.
DTI drug target interaction.

ML machine learning.

PCA Principal Component Analysis.

QSAR quantitative structure-activity relationship.

ReLU rectified linear unit.
RF random forest.

SMILE simplified-input line-entry.
SVM support vector machine.

TF TensorFlow.
TPR true positive ratio.

List of Figures

2.1 Main steps of the drug discovery flow 7
2.2 Process of Drug Repositioning 8
2.3 A comparison of traditional *de novo* drug discovery and development *versus* drug repositioning 9
2.4 Areas of computer-aided drug discovery 10
2.5 Artificial intelligence, machine learning and deep learning 15
2.6 Machine learning as a new programming paradigm 16
2.7 Flowcharts about different AI disciplines 18
2.8 Deep learning and its data transformation 19
2.9 Scheme of a perceptron 20
2.10 Generic neural network 21
2.11 Scheme of Deep Learning flow 22

3.1 Data set construction 32
3.2 Scheme of k-fold cross validation 35
3.3 Deep feed forward network concepts 38
3.4 Geometrical view of an error function 39
3.5 Neural network step-by-step 41

4.1 2D visualization of the data set 48
4.2 Baseline architecture 49
4.3 Model performance according to the number of epochs 52
4.4 The flowchart of the proposed deep learning pipeline 55

List of Tables

4.1 Different types of descriptors for targets 47
4.2 Different types of descriptors for drugs 47
4.3 Dispersion of feature 700 between the two classes 49
4.4 Set of possible values for number of epochs, batch-size and optimizer used on the grid-search 52
4.5 Grid-search results for number of epochs, batch-size and optimize . . 53
4.6 Final parameters used on the model constructed 54
4.7 Trainable parameters (weights and biases) for the model 54
4.8 Evaluation metrics results for 100 runs 56
4.9 Confusion matrix for one run 56

Contents

List of Acronyms **xi**

List of Figures **xiii**

List of Tables **xv**

1 Introduction **1**

1.1 Context . . . 1
1.2 Motivation . . . 2
1.3 Goals . . . 3
1.4 Research Contributions . . . 4
1.5 Document Structure . . . 4

2 State of the art **5**

2.1 Drug discovery . . . 5
2.1.1 Drug discovery process . . . 6
2.1.2 Drug repositioning . . . 7
2.1.3 Computer aided drug design . . . 10
2.1.4 Drug target interaction prediction . . . 11
2.2 Deep learning . . . 14
2.2.1 Artificial intelligence . . . 15
2.2.2 Machine Learning . . . 16
2.2.3 The ”deep” in deep learning . . . 17
2.2.4 How deep learning works - neural networks . . . 19
2.2.5 Deep learning applications . . . 23
2.2.6 The rise of Deep Learning for Drug Discovery . . . 24

3 Methods **29**

3.1 Data preparation . . . 29
3.1.1 Positive data set construction . . . 29

3.1.2 Negative data set construction . . . 30
3.1.3 Final data set construction . . . 31
3.1.4 DTI descriptors . . . 31
3.2 Data analysis . . . 32
3.2.1 Data exploration . . . 32
3.2.2 Data pre-processing . . . 33
3.2.3 Data visualization . . . 33
3.2.4 Data splitting . . . 34
3.2.4.1 K-fold cross validation . . . 34
3.3 Feature engineering . . . 34
3.4 Classification model implementation . . . 36
3.4.1 Frameworks . . . 36
3.4.2 Deep feed forward network . . . 36
3.4.3 Parameter optimization . . . 38
3.4.4 Hyper-parameter tuning . . . 41
3.4.4.1 Grid-search . . . 42
3.4.5 Model improvement . . . 43
3.5 Evaluation metrics . . . 43

4 Results and Discussion **47**
4.1 Drug target space treatment . . . 47
4.2 Baseline approach . . . 49
4.3 Architecture decisions . . . 50
4.4 Network regularization . . . 54
4.5 Evaluation metrics . . . 56
4.6 Final remarks . . . 57

5 Conclusion **61**

Bibliography **63**

1

Introduction

1.1 Context

One of the biggest medical breakthroughs of the twentieth century was the discovery of antibiotics, which was immediately followed by the unfortunate emergence of bacterial antibiotic resistance [1]. Nowadays antibacterial resistance is becoming more frequent and is a major concern that many are trying to address. Since 2012, resistance to multiple antibiotics has been steadily increasing and its already high globally, being considered as an emergent global disease and one of the main public health problems [2, 3]. Development of antibacterial resistance is the result of a cascade of events triggered by continued selective pressure, which creates the ideal scenario for the rapid dissemination of resistant microorganisms and horizontal transfer of resistance genes.

Furthermore, in the long run, antibacterial resistance also have a financial negative impact. The annual economic burden associated with the treatment of antibiotic resistant infections has been estimated to be between $21,000 and $34,000 million in the United States alone, and around €15,000 million in Europe [4].

The current global threat of antimicrobial resistance and the urgent need to control it and to find new antibacterial products has prompted the different stakeholders to take action in integrating research and public health. There is a need to circumvent the current situation and create measures to achieve this goal. Besides the need of all the stakeholders being involved in order to correct the current situation, the task falls on the shoulders of academia due to the fact that pharmaceutical industry has ceased investing in antibiotic discovery owing to high cost, lengthening developing cycles, complexities and low profits. Policy makers, public health authorities, regulatory agencies, pharmaceutical companies, all should work together to achieve strong regulatory modifications and revert de situation.

Thus we are facing a major medical and pharmaceutical challenge. Today, more than ever, is necessary to develop new antibiotics or drugs able to neutralize antimicrobial resistant pathogens.

1.2 Motivation

The traditional pipeline for drug discovery and deployment is very complex and slow. The entire process, from drug conceptualization to market entrance, can take up to 15 years and reach a total cost of $1,000 million. Furthermore, there is no warranty the identified compound will enter the market and generate revenue.

The drug discovery process depends heavily on its first three stages: target identification, lead discovery, and lead optimization. One of the crucial steps in narrow-spectrum antibiotics development is target identification [5].

In the pharmaceutical industry, early to mid phase drug discovery efforts concentrate on advancing therapeutically relevant small molecules and bringing candidate compounds into clinical trials. In order to be a good candidate, a drug target must be efficacious in blocking or promoting a biological event of interest, safe and meet clinical and commercial needs [6].

Drugs fail in the clinic for two main reasons; the first is that they do not work and the second is that they are not safe. As such, one of the most important steps in developing a new drug is target identification and validation [7].

Computational methods are mostly, but not exclusively, applied during the early phase of drug discovery. This computational analysis help on the decision making process, allowing to reduce the number of candidates compounds to be evaluated experimentally. Many computational techniques have been applied to this field of study improving substantially the entire drug discovery process. Since new technologies and computational approaches are always emerging, such as Deep Learning (DL), there is a need to understand how this techniques would behave on this particular field.

Deep learning has been successfully used on many topics. Drug discovery, despite the use of many computational techniques on the process, is still lacking on models capable of understanding precisely biological data. The fact that only very few deep architecture were applied to drug discovery, should be motivational enough to study it.

1.3 Goals

The main goal of this work is to study new computational techniques, in particularly deep learning, to the field of drug discovery, more specifically on the drug target interaction (DTI) prediction.

Deep learning is an emerging computational technique, which has became state-of-the-art in many subjects, such as image or speech recognition. On the field of DTI, other techniques have been applied, such as tree based algorithms which are more traditional methods. There is still room to improve this kind of prediction with DL, since these type of techniques have been improving research on other fields.

The goal of this work is to understand how DL techniques behave on DTI data when compared to standard methods. Try different models, explore among many possible architectures, assess them towards external data and picking and build the best one, are also objectives to achieve.

DL represents a broad class of techniques, in which, one of the main challenges in applying them is selecting the appropriate architecture for a specific task. It requires a complete study of the data available and an understanding of its statistical behaviour, in order to choose the right architecture. Furthermore, deep learning is a hot topic so, frequently, there are new hypothesis showing up, which makes it hard to be fully contextualized about the state-of-the-art of such techniques.

To achieve this goal we had to learn DL from scratch, dive into this exciting field, understand how it works, what models could be applied to our problem and build them.

At the end, we expect to create a pipeline for DTI prediction, using deep learning techniques. Applying some feature engineering and choose the right parameters will help to build the best model, which will be compared to other works on this field. Furthermore, the capability of generalization of the model must be understood, in order to conclude about the future usage of this recent techniques on the field of drug discovery.

With the pipeline complete, we want to prove DL effectiveness on DTI challenge and understand if traditional methodologies already applied are outperformed. Another main goal is to reinforce the need for deep learning study on other fields, such as drug discovery, which is a new topic for many researchers and biologists.

To sum up, on the course of this work we want to find answers for the following

questions:

- What is the value of using deep learning approaches on drug discovery process?
- Which is the best deep learning architecture to perform drug target interaction prediction?
- Should the academia and companies focus on developing deep learning based solutions for drug discovery?

1.4 Research Contributions

The work developed during this thesis resulted in the following contribution:

- Guilherme P. Coelho, Joel P. Arrais, Bernardete M. Ribeiro. "Deep Learning for Drug Target Interaction Prediction". RECPAD 2018, 24th Portuguese Conference on Pattern Recognition (submitted).

1.5 Document Structure

The remainder of this document is organised as follows: in Chapter 2 we provide some useful information related to the subjects that are the basis of this work along with several research works from the drug discovery field. Then, Chapter 3 describes the methods carried out on to perform this work and Chapter 4 presents the obtained results as well as its discussion. Finally, Chapter 5 concludes the thesis and presents some possibilities for future work.

2

State of the art

In this section we will go into further detail on the topic of drug discovery, with a particular focus on the drug target interaction subject. Later we will talk about deep learning and understand its current state. To finish this chapter the possibilities of applying deep learning techniques to drug discovery will be analysed.

2.1 Drug discovery

The discovery of novel drug targets is a huge challenge in drug development. Although the human genome comprises approximately 30,000 genes, proteins encoded by fewer than 400 are used as drug targets in the treatment of diseases [8].

To date, the discovery and development of new drugs have relied heavily on the use of preclinical animal models. It is widely recognised that this reliance on animal models, does not represent effectively the human systems behaviour, which leads to some limitations on the drug discovery process [9]. Besides that, computational approaches are an integral part of the interdisciplinary drug discovery research. The truth is that remains weakness in the methods developed so far [7]. The bottom line is if computational approaches are capable to work around the issue related to using animal models, and help on increasing the efficiency on this obsolescent drug discovery process.

Furthermore, when a candidate drug enters clinical trials there is a probability of 90% that, later, it fails to demonstrate sufficient safety and efficacy to gain the regulatory approval needed. This happens mainly as a result of insufficient efficacy and/or unacceptable toxicity, due to the limited predictive value of preclinical studies [10]. Although new drugs approval reached a 66-year high in 2015, a new 6-year low was reached in 2016 resulting on the approval of only 22 new medicines [11] later released to the end user. This amount of failures are expensive and indicate our

inability to entirely characterize the efficacy or to realize the potential liabilities of candidate compounds. The process is lacking of a method to mislead candidates that later will fail.

Understandably, the traditional drug discovery and development process is complex, time and money consuming. Fortunately, the experimental efforts over the past years, allowed the compilation of public databases, which enhanced the probability of developing efficient computational methods [12]. For this reason, computational methods could improve significantly this complex process.

2.1.1 Drug discovery process

A drug discovery programme starts because there is a clinical condition without a suitable and proper medical product available. This undiscovered clinical need is the underlying driving motivation for the process. *De novo* experimental drug discovery is composed by a series of processes which the outcome is the identification of the drug compounds for a specific treatment or control of disease target. It starts with the screening a large number of chemical compounds to optimize the disease targets. In order to perform this, we need insight information about the structure of drug receptor (target).

Drug discovery process begins with the understanding of the disease. We must understand what are the causes and what biological mechanisms are related to it. Only by this way we will be able to design a drug to target that specific disease. The process consists on the following steps [13]:

1. Candidate drug discovery
    - Target identification
    - Lead discovery
    - Lead optimization
2. Pre clinical and clinical trials to evaluate the safety, efficacy and adverse effects of the drug
    - Animal studies
    - Clinical trials
3. Regulatory agencies approval process for the newly discovered drug and bringing the drug to market for public use

- Additional post marketing testing
- Further improvement of the drug

In Figure 2.1 we can see a diagram of the main components of the drug discovery process.

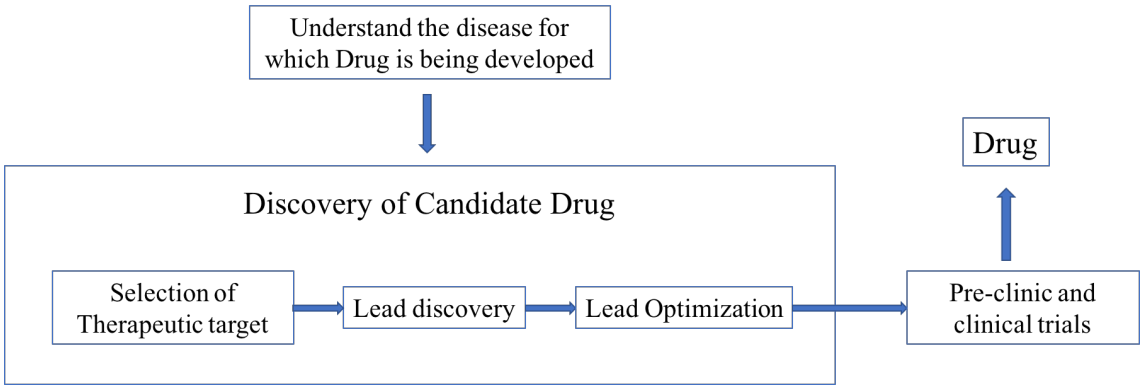

Figure 2.1: Main steps of the drug discovery flow. Adapted from Pratik Swarup Das *et al.* [13].

As previously stated, one of the most important steps in developing a new drug is target identification and validation [6]. A target is a broad term which can be applied to a range of biological entities which may include for example proteins, genes (DNA) and RNA. These entities vary according to the disease or clinical condition that is been studied. A good target needs to be, above all, 'druggable'. A 'druggable' target is accessible to the putative drug molecules, which means both entities will interact and create the desired output.

This work will focus on the first step, candidate drug discovery. Besides being a extremely relevant phase, it is the one where new computational methods can substantially improve all the efforts behind it.

2.1.2 Drug repositioning

Biopharmaceutical companies attempting to increase productivity on drug discovery have been investing in novel discovery techniques. Such techniques have failed to achieving the desired results. Actually, novel and promising techniques, like the widely used structure-based drug design, combinatorial chemistry, high-throughput screening (HTS) and genomics, have failed to delivery the improvement of productivity expected. This problem has forced the drug developers to search for innovative

ways of redo the process, such as finding new uses for existing drugs or create improved versions of them [14]. This is called drug repositioning.

In recent years, drug repositioning or drug repurposing has become an increasingly popular trend in drug discovery.

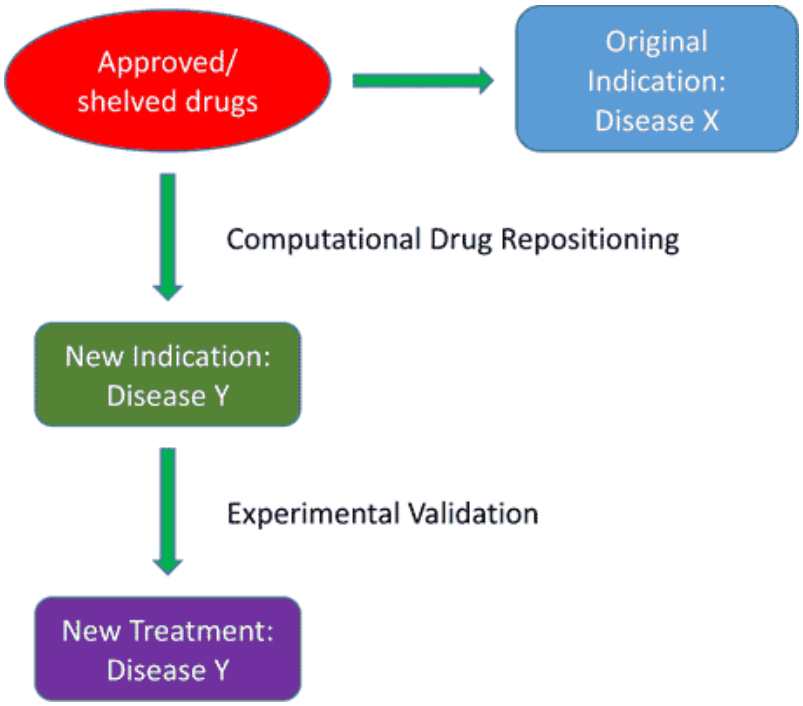

Figure 2.2: Process of Drug Repositioning. When an approved drug causes a different effect from the original, there is a possibility to turn these into main effects and generate a new use for that specific drug, which will later be submitted to experimental validation [15].

By focusing on one of the undesired effects of an already commercialized drug in an attempt to make it the main effect, it is possible to reposition that drug for new uses, as shown in Figure 2.2. Recent literature reveals that many drugs often possess the so-called *promiscuity* property [16]. This property represents the drug ability to act on other off-target proteins in addition to the original target. This theoretical evidence provides a strong support for drug repositioning.

There are several examples of successfully repositioned drugs for uses different from their original indications. *Sildenafil* is probably the most popular example, which was initially used to treat hypertension, then angina, and currently is used for erectile dysfunction [17, 16]. Drug repositioning can be achieve by different strategies, from using computational methodologies to exploring the micro-organism genome information. Computational drug repurposing approaches, invariably make use of previously known drug-target associations, which is the methodology under study on

this work. Such computational methodologies allow rapid and inexpensive screening of a broad spectrum of drugs and targets, either by screening ligands for a certain drug-target, or screening potential drug-targets for a specific ligand, with the objective of finding alternative targets for known drugs.

Repositioning existing drugs for new indications could deliver the productivity increases that the industry needs. This method have low risk associated due to the fact that repositioning candidates have often been through several stages of clinical development and therefore have well-known safety and pharmacokinetic profiles, which will reduce the probability of not getting the final approval to enter the market. Furthermore, by using this method it is possible to create shorter route to the clinic, because many phases, such as chemical optimization, toxicology, have, in many cases, already been completed and can, therefore, be bypassed. To sum, drug repositioning allows substantial reduction of risk and costs on the pathway of drug discovery.

In Figure 2.3 we can understand the differences of *de novo* drug discovery and development when compared to drug repositioning strategy.

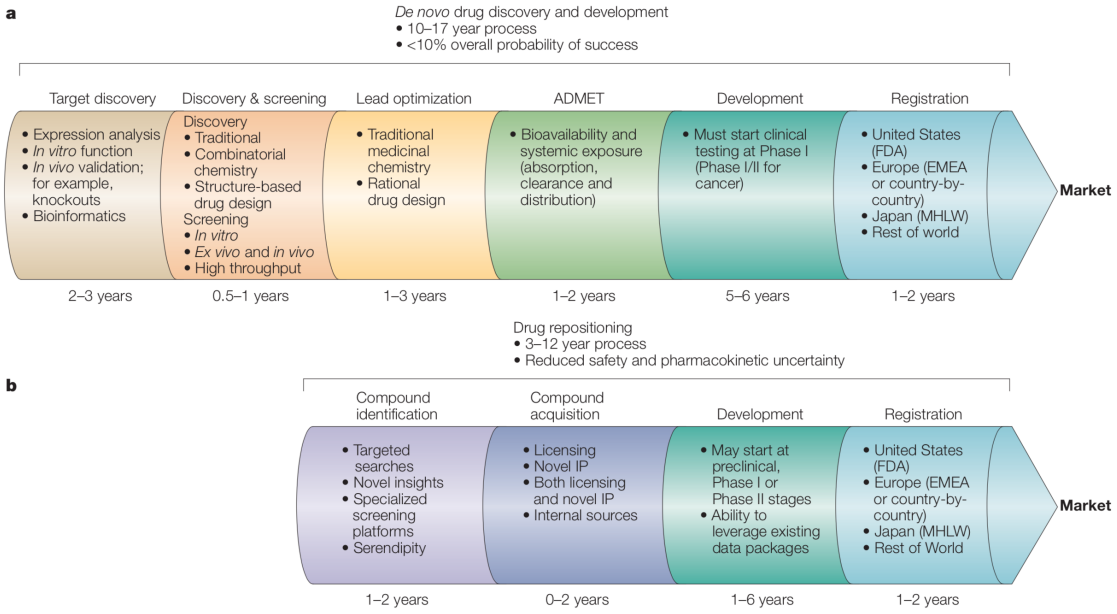

Figure 2.3: A comparison of traditional de novo drug discovery and development versus drug repositioning. **a:** It is well known that *de novo* drug discovery and development may take 10 to 17 years, from idea to market, and has a success rate lower than 10%. **b:** Drug repositioning can reduce time and risk as several common phases can be bypassed [14].

Drug repositioning have been emerging in recent years and the interest on it has been growing, both from pharmaceutical or biotech companies and venture capitalist [14].

Its advantages are recognized and the activity on the area has increased dramatically. The unique challenge is related to the repositioning strategies, which demand for creative approaches. Once again innovative computational methods could help on this repositioning exploratory task. On this work we will, precisely test new technical approaches to help on drug repositioning.

2.1.3 Computer aided drug design

Discovery and developing a new medicine is a long, complex, costly and highly risky process that has few peers in the commercial world. This is why computer aided drug design (CADD) approaches are being widely used in the pharmaceutical industry to accelerate the process. In Figure 2.4 we see in which areas of drug discovery process, computer-aided methods are involved. Various computational approaches are already employed in this, highly complex and resource intense, process. CADD provides several tools and techniques that help in various stages of drug design.

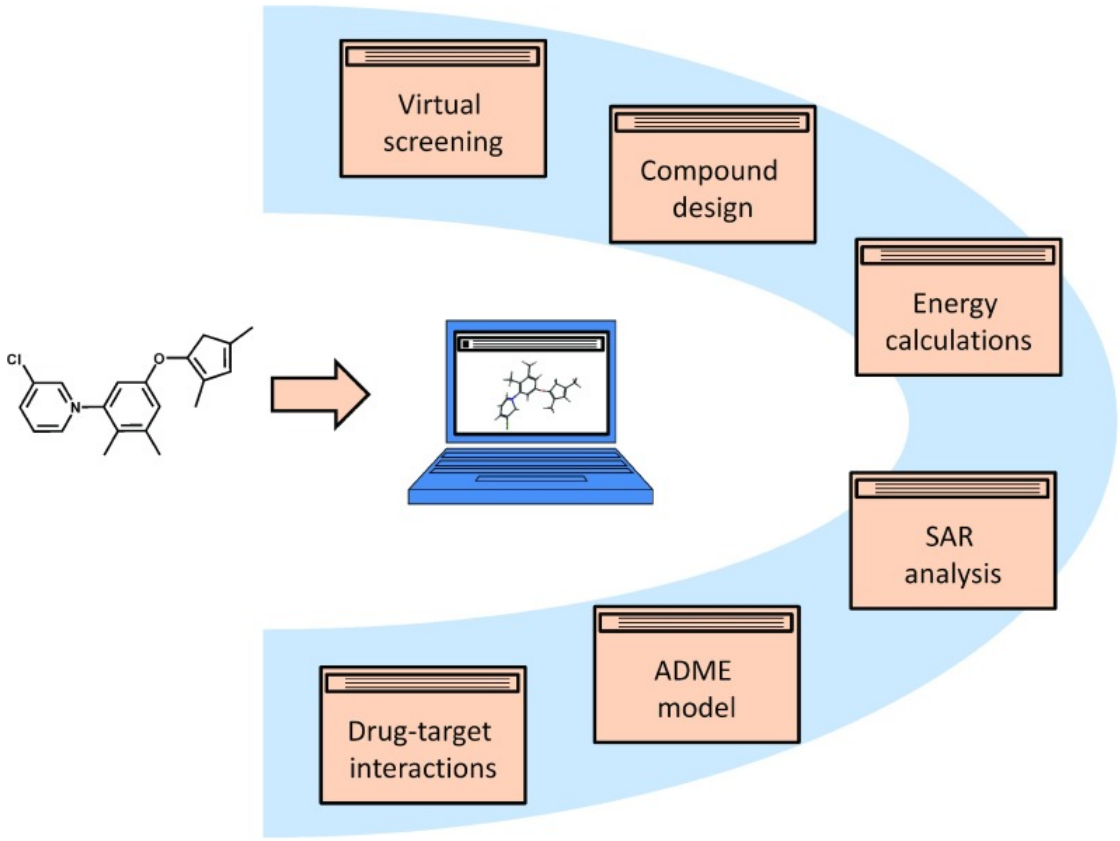

Figure 2.4: Areas of computer-aided drug discovery. Computer aided approaches are being using on several phases of the drug discovery flow [7]. ADME - absorption, distribution, metabolism, and excretion; SAR - structure-activity relationship.

Only due to the amount of data gathered throughout the years with *in vivo* experiments is possible to accurately develop and implement the referred computational methodologies. If adjusted correctly, these methods improve the ability to identify and evaluate potential drug molecules. Focusing on the identification of drug target interactions towards targets of biological interest, either to stop or enhance a determined molecular function, dedicated models for protein simulation and DTI prediction are crucial to speed up the actual process and to reduce the costs associated with it.

Data mining of available biomedical data has led to a significant increase in discovery candidate drugs. On this particular topic, data mining refers to the use of an informatic approach to help identifying, selecting and prioritize potential drugs. The available data comes from a variety of sources and contains different types of information related to the biological entities under study. Machine learning (ML) and data mining methods have been widely used on the computational biology and bioinformatics area.

The growing ubiquity of predictive models across drug discovery phases, from target identification to the clinical set, has helped to significantly change the current situation.

2.1.4 Drug target interaction prediction

As previously stated, computational methods are mostly, but not exclusively, used on the early stage of drug discovery. On this phase the main efforts are to deciphering disease-related biology, prioritizing drug targets, and identifying and optimize new chemical entities for therapeutic intervention [7].

Specifically the systematic assessment of drug target interactions is one the areas that have, recently, witnessed a great improvement, due the amount of data available. *In silico* prediction of unknown DTIs has become a popular tool for drug repositioning and drug development. On this subject, predictive models of target activity can be derived on the basis of compound activity classes using machine learning methods. Systematic accounts of ligand target interactions have made it possible to predict unwanted side effects of drugs or candidates [18].

There is a need to reduce the number os candidates compounds that are evaluated experimentally, which can be achieve with computational analysis. This analysis may provide essential help when it comes to decide which are the candidate drugs.

Since the failure rates on clinical trial are extremely high, the major challenge is trying to pick the best possible candidates and consequently reduce their number and the failure rates.

Dedicated computational methods for DTI are crucial for speed up and reduce the costs associated with drug discovery, because they allow to select new leads for drug repositioning [19]. Identifying the interactions between drugs and target proteins is a key step in drug discovery. This not only aids to understand the disease mechanism, but also helps to identify unexpected therapeutic activity or adverse side effects of drugs.

Nowadays, many *in silico* approaches have been developed to identify new drug target interaction. Ligand-based and structure-based are two of the most used computational approaches for this field. Related to ligand-based methodology, many carried out the strategy of applying quantitative structure-activity relationship (QSAR) to predict the bioactivity of a molecule on a target. QSAR is based on the hypothesis that molecules with similar structure have similar bioactivity [20]. Despite all the times is was successfully used, the performance of these models decreases if the number of known active molecules of a target is not enough. Furthermore with each QSAR model we can only predict activity against one solely target. These disadvantages led the researchers to shift to other alternatives. Structure-based methods can only be applied when the three-dimensional (3D) structure of a target is known and available resulting on molecular docking.

Yang *et al.* [21], in an attempt to address the DTI prediction problem of being mainly focused on relate genomic and chemical data, systematically integrate large-scale chemical, pharmacological, genomic and functional data and DTI network information into a unified framework. This led to the implementation of a conditional random field (CRF) method, which integrates all that information. The CRF is a probabilistic graph model capable of encoding the drug target network for DTI prediction. To train the model, they apply a stochastic gradient ascent approach and the contrastive divergence algorithm. At the end they could identify the hidden associations between drugs and targets [21]. The tests performed to evaluate the model show that it could achieve excellent prediction performance with the area under the precision-recall curve (AUPR) up to 0.949. The approaches performed on this work might have potential when applied to reposition certain drugs. Although using functional similarity data might dismiss its use for drug repositioning in infectious diseases [19]. The most likely result of using functional similarity for this purpose would be the prediction of an antibacterial drug that would continue the

selective pressure in the target microorganism, perpetuating antibacterial resistance.

In the work by Cheng *et al.* [22], three inference methods were developed to predict new DTI: drug based similarity inference (DBSI), target-based similarity inference (TBSI) and network-based inference (NBI). Four benchmark data sets were used to assess the performance of all the methods and NBI always end up with the best evaluation values, suggesting that it would be the one with highest predictive ability. The network-based inference method, uses known drug target bipartite network topology similarity to predict new and unknown DTI. Some of their predictions were validated by in vitro assays, confirming that five drugs had pharmacological effects on different and alternative targets. The average area under ROC curve (AUC) value of NBI method by the 30 simulation times of 10-fold cross validation, among all biological entities, was 0.934 [22]. The biggest disadvantage of NBI is that it could not be applied to the new drugs without any known target information in the training set.

He *et al.* [23] also developed a different approach for *in silico* predictions of DTI. To achieve this, drug compounds were encoded with functional groups and proteins by biological features including biochemical and physicochemical properties. They have created a positive dataset from public information of experimental trials. The corresponding negative data set was derived from the above positive datasets via the following steps: (1) separate the pairs on the positive dataset into single drugs and proteins; (2) re-couple these singles into pairs in a avoiding repeating them on the positive dataset; (3) randomly picked the negative pairs thus formed until they reached the number two times as many as the positive pairs. The representation of drug was achieved by selecting 28 common functional groups. Proteins description was formulated as a 139-D vector, according to their components. With all samples represented by a feature vector, a predictor using a machine learning approach was developed. The nearest neighbour) algorithm is quite popular in pattern recognition community, and so was the algorithm chosen. They soon realized a better feature set could improve prediction performance. Some features could be correlated and others did not bring relevant information for the prediction power. After using maximum relevance minimum redundancy to do feature evaluation, the dataset was smaller. Although there were more feature for target than for drug, more drug features came as a result of feature evaluation, showing the important role of drugs. Many features selected for the protein were related to its secondary structure. Instead of classifying the proteins as a whole family, target proteins were divided into four groups: enzymes, ion channels, G-protein- coupled receptors and nuclear receptors. At the end, the overall success rates of cross-validation tests achieved with the four

predictors were 85.48%, 80.78%, 78.49%, and 85.66%, respectively [23].

Coelho *et al.* approach the DTI challenge with a random forest (RF) classification model [19]. They developed a computational pipeline able to discover putative leads for drug repositioning that could be applied to any microbial proteome. After a five-fold cross-validation for internal validation and a test of external data, 99% and 91% was achieved for the AUCs, respectively. This approach was based solely on the primary sequence of the protein and the simplified-input line-entry (SMILE) of the ligand. In addition molecular docking experiments were performed to validate the DTI predictions obtained. The docking results indicated the validity of the proposed architecture.

On more recent works, novel computational approaches have been developed. Particularly on the work by Wang *et al.* [24], a model for predicting DTIs was created under the theory that each pair could be represented by the structural properties from drugs and evolutionary information derived from proteins. To achieve this representation, protein sequences were encoded as position-specific scoring matrix descriptor, which contains the information of biological evolutionary. Similarly to other works drugs were encoded by representations of specific functional groups of fragments. The prediction model was established with rotation forest model and applied on four datasets, enzymes, ion channels, G protein-coupled receptors and nuclear receptors. The proposed architecture achieved a prediction accuracy (ACC) of 91.3%, 89.1%, 84.1% and 71.1% for four datasets respectively [24].

This work focus on applying innovative computational methodologies particularly to the drug target interaction phase, which will be a complement to the decision making process of picking the best drug candidates and for drug repositioning. The main goal is to predict whether or not a specific drug will positively interact with a target, based only on the primary sequence of the protein and the SMILE system of the ligand.

2.2 Deep learning

Inventors have long dreamed of creating machines that think. This desire dates back to at least the time of ancient Greece [25]. Digital data, in all shapes and sizes, is growing exponentially, everyone have probably heard about big data and about the explosion of electronic devices with tremendous computational power. The fact that almost every process in our world uses kind of software is giving us massive amounts

of data every minute. The high demand of exploring and analysing big data has encouraged the use of data-hungry machine learning algorithms like deep learning.

In the past few years, artificial intelligence (AI) has been a subject of intense media hype. Machine learning, deep learning and AI come up in countless articles, mainly due to the amount of data that recently become available.

Deep learning is a particular field of machine learning and consequently of artificial intelligence. DL is a new approach to learn representation of given data. These techniques represent a remarkable step forward taken by machine learning, in recent decades, with, never seen results, on several topics, and as previously stated will be the computational techniques explored for the field of drug discovery on this work.

2.2.1 Artificial intelligence

Artificial Intelligence was born in the 1950s by the creative minds of a group of pioneers who wonder whether computers could be made to ”think”. A concise definition of this field would be *”the effort to automate intellectual task normally performed by humans”*. As such, AI is a wider subject that includes both machine learning and deep learning, as shown in Figure 2.5.

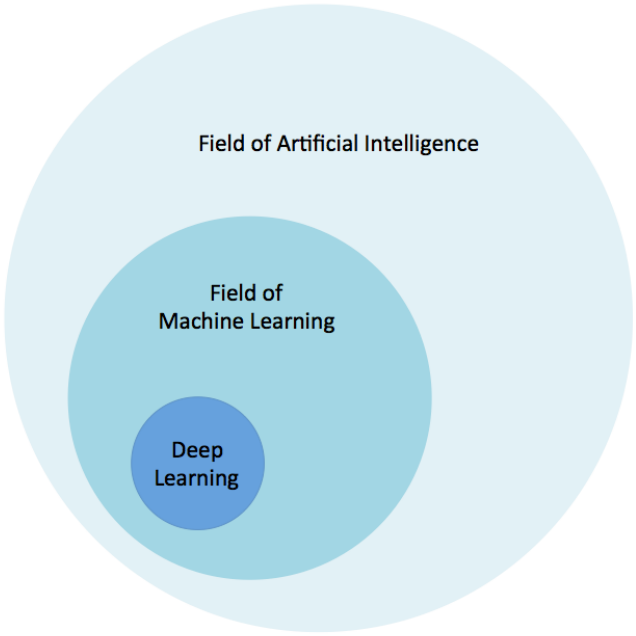

Figure 2.5: Artificial intelligence, machine learning and deep learning.

The term is frequently applied to the project of developing systems with the same

intellectual characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience [26].

2.2.2 Machine Learning

Machine learning is a sub-branch of AI, and even among machine learning practitioners there is not a well accepted definition of what is and what is not machine learning. Machine learning represents the ability of a computer to go beyond "what we know how to order it to perform" and learn by its own how to perform so it can accomplish a specific task.

Machine learning means that, rather than programmers defining rule by hand the computer will automatically learn these rule by just looking at the data. Machine learning allows, in opposite to classical program, to give data and answers as input and expect the rules as an outcome. In Figure 2.6 we can see these two opposite programming strategies.

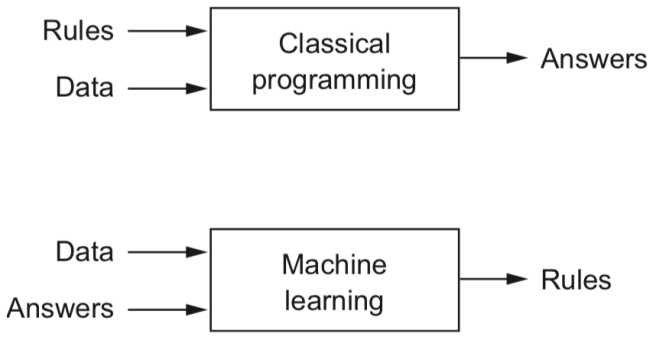

Figure 2.6: Machine learning as a new programming paradigm.

Machine Learning is actually a current application of AI based around the idea that we should really just be able to give machines access to data and let them learn for themselves. A machine learning algorithm is trained instead of explicitly programmed. The algorithm takes as an input many examples relevant to a task, and finds patterns within the data receive. This discovered patterns will eventually allow the system to come up with rules for automating the task.

A system based on machine learning, may, on the basis of the observation of previously processed data, improve its knowledge, so it can, in the future, achieve better results or, produce an output closer to the one expected for that specific problem [27].

Although machine learning only started to flourish in the 1990s it quickly deserved the attention of many enthusiasts. Nowadays, machine learning is the most popular and successful subject of AI.

2.2.3 The ”deep” in deep learning

There are many reasons that led to deep learning being developed and becoming the centre of machine learning in the recent years. The first, and perhaps the main one, is related to the progress we have witnessed in hardware. Mainly, graphics processing units (GPU) contribute to making possible to training deep learning networks. GPU's have greatly reduced the time needed to run a this kind of networks, lowering them by a factor of 10 to 20 [27]. Other reason focus on what we have already stated, the amount of data that become available recently. We have numerous datasets, with a high dimensionality as an input data, and a technique capable of dealing with it was needed. So, facilitated by the intersection of inexpensive computing power, unprecedented large data sets, and clever computational statistics advances, deep learning algorithms are driving an artificial intelligence revolution.

Many tasks on the machine learning field can be solved by just designing the appropriate feature space and giving them to a machine learning algorithm. Although there are other task in which the difficulty to understand the features that have a strong predictive power and should be extracted becomes a problem. For example, if we want to detect a car on a picture we might like to use the presence of a wheel as a feature. Disappointingly, in terms of pixel values, it is even a bigger problem to describe how a wheel looks like. One solution to this is create algorithms capable of discovering not only, the mapping to turn input data into the desired output, as on machine learning methods, but also the representation of the input data itself [25]. This innovative approach to the problem is known as representation learning. These new methodologies often result in much better performance when compared to hand-designed representation. Deep learning belongs to the representation learning set of methods and solves this issue of data representation. In Figure 2.7 are shown different flowcharts, where is possible to understand the contrast between Deep Learning and other machine learning methods in terms of data representation.

Deep learning is based on the way the human brain process the information and learn to respond to certain stimuli. This recent approach is related to a learning method based on successive layers of increasingly meaningful representations for the data. This methodologies uses artificial neural networks (ANN) which contains

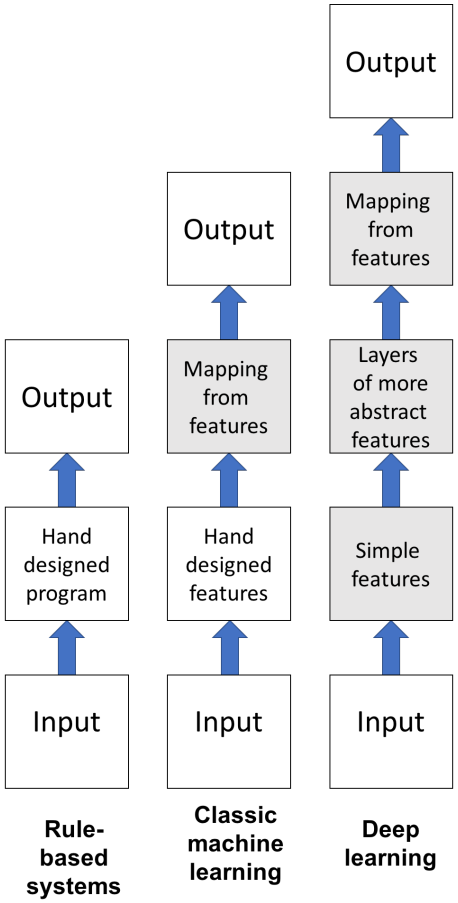

Figure 2.7: Flowcharts showing how the various AI disciplines related between each other. Shaded boxes indicate components which can be learned from raw data. Adapted from Goodfellow *et al.* [25].

many layers, the reason behind the word *deep*. Curiously this meaning has changed with time. While 4 years ago, 10 layers were enough to call a network *deep*, nowadays when we refer to deep learning we are talking about hundreds of layers [28].

So, deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features [29]. By learning features at multiples levels of abstraction it is possible for a system to create complex functions and relate the input data with the desired output, developing a map between them, without depending completely on human-crafted features. From the raw input image data into the desired output, which is its description, data is transformed, layer by layer, into gradually higher levels of representation representing more and more abstract functions of the input data, e.g., edges, local shapes, object parts, among others. In practice, the "right" representation for all these levels of abstractions is not known, we have to trust the model with this task. Deep learning enables the computer to build complex concepts out of simpler con-

cepts. In Figure 2.8 we can see an example of this feature construction carry out by a deep learning method.

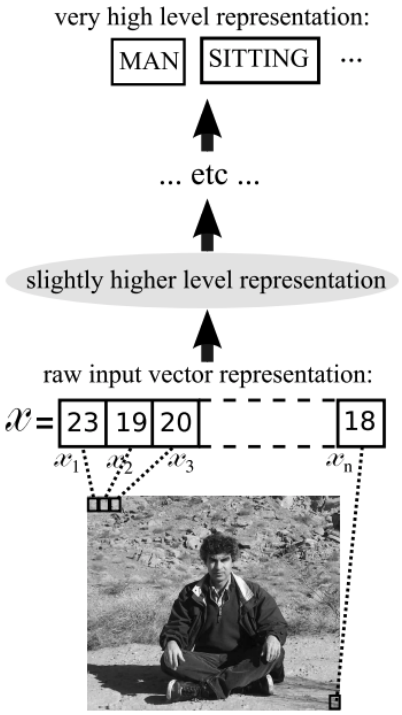

Figure 2.8: The transformation of raw input data into gradually higher levels of its representation [29].

This ability of automatically learn powerful features will become increasingly important as the amount of data and range of applications to machine learning methods continues to grow [30].

As stated, in deep learning these layered representation are learned through models called neural networks, structured in literal layers stacked on top of each other. The term neural networks is a reference to neurobiology. In faction some important fact behind deep learning were inspired from our understanding of how the human brain works.

2.2.4 How deep learning works - neural networks

The development of neural networks has been fundamental to get closer to the goal of teaching computers to think and understand the world in the way we, humans, do, while retaining the innate advantages they hold over us such as speed, accuracy and lack of bias [31].

Deep learning involves stacking these straightforward little algorithms called artificial neurons together to solve problems. Neurons are the basic unit of a deep learning model. The first artificial neuron developed was a perceptron and was created by the scientist Frank Rosenblatt. A perceptron is a unit that takes several binary inputs, x_1,x_2,x_3,..., and produces a single binary output.

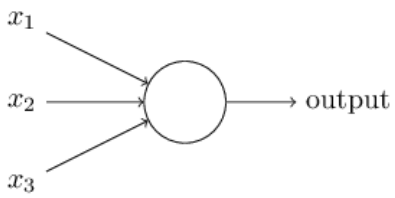

Figure 2.9: Scheme of a perceptron.

Rosenblatt proposed a simple rule to compute the output. He introduced weights,w_1,w_2..., real numbers expressing the importance of the respective inputs to the output. The neuron's output is binary, either 0 or 1 and is determined by whether the weighted sum $\sum_j w_i x_i$ is less than or greater than a specific threshold value, as shown on 2.1. The threshold, used on the other side of the function becomes the perceptron bias. Today, it's more common to use other and more complex models of artificial neurons. The most used neuron model is one called the sigmoid neuron.

$$output = \begin{cases} 0, & \text{if } \sum_j w_i x_i \leq \text{threshold} \\ 1, & \text{otherwise.} \end{cases} \tag{2.1}$$

By combining many artificial neurons, on many layer and with different weights and thresholds we can get diverse neural networks. Deep learning is, in fact, a class of neural networks characterized by a significant number of layers of neurons, which gain the ability to learn rather complex models, based on progressive levels of abstraction.

If a network has many layers and neurons, it is called multilayer perceptron (MLP), which is a generic neural network, such as in Figure 2.10. Every network has an input layer, one or more intermediate layer, also known as hidden layers and the output layer. As on the perceptron, on this network we also have weighs. A mathematical function called the activation function is then applied to the sum to form the new value of the neuron. The specification of what a layer does to the data it receives it stored on the layer weights, which is, no more, than a bunch of numbers. The transformation performed on each layer is parameterize by its weights. On this

context, learning means to achieve the best values for the weights of all the layers of the network. With this set os values the network will be able to correctly map input data into de desired output.

At the beginning of training task, we have no clue about what should be the best values for the weights and so these values are randomly selected. Although it must be a way to control them and understand if we are getting closer to all goal of achieve the best weights to represent our data. In fact, there is a method to measure how far the output is from what we expected, and consequently to measure the performance of the set of weights. This is the job of loss function of the network. This functions takes the predictions of the network and the true target and computes a distance score. This score represents how well the network has performed.

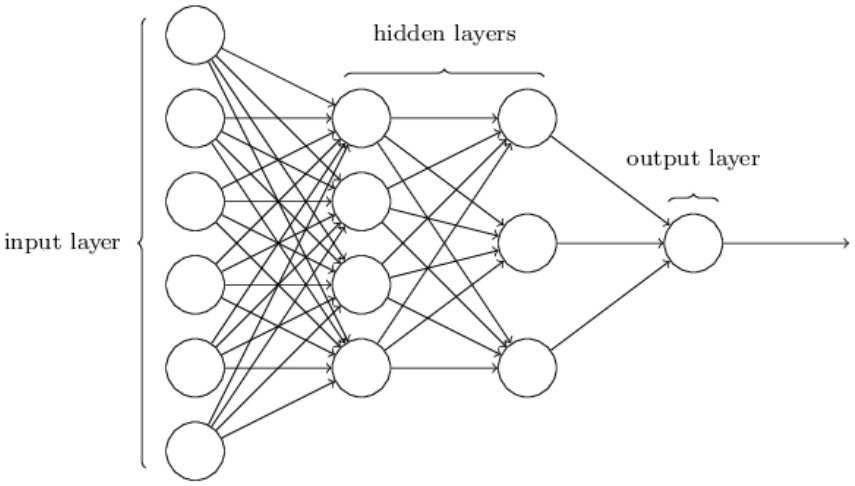

Figure 2.10: Generic neural network.

The fundamental trick in deep learning is to use this score as a feedback signal, and accordingly to it, adjust the network parameters in such way, that the score of the loss function will decrease [32]. This small adjustment is the job of the optimizer which implements what is called the backpropagation algorithm. This concepts are summarized in Figure 2.11.

Initially, the weights of the network are randomly setted and so it is merely a series of random transformations. Naturally, at this point, the output is far from the expected one, and consequently the score from the loss function will be higher. Although, iteration after iteration the weights will be adjusted a little into the correct direction and the loss function will decrease. To properly adjust the weight vector, the learning algorithm computes a gradient vector that, for each weight, indicates by what amount the error would be affected, of the weight were increased by a tiny amount. To this amount, by which the weight is changed, we give the

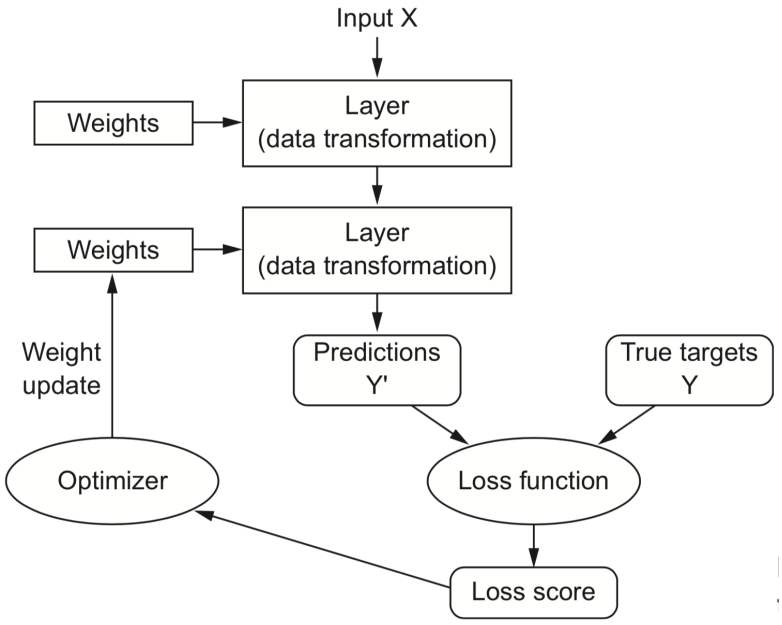

Figure 2.11: Schema of Deep Learning flow. The loss score is used as a feedback signal to adjust the weights [32].

name on learning rate. In practice, many developers are using a procedure called stochastic gradient descent (SDG) [30]. This is called the training loop, which, repeated many times, will minimize the loss function and optimize our output.

But the traditional artificial neural network (ANN) method suffered from problems such as overfitting, diminishing gradients, among others, and was largely replaced by other machine learning algorithms like support vector machine (SVM) and random forest. The recent development of DL has given ANN a renaissance. The major difference between DL and traditional ANN is the scale and complexity of the NNs. DL uses larger numbers of hidden layers whereas traditional ANNs normally can only afford one or two hidden layers owing to the limitation of computer hardware in the early days.

There are also many algorithmic improvements in DL, for example using the dropout and earlystop methods to address the overfitting problem or applying rectified linear unit (ReLU) to avoid vanishing gradients, for example. With the rise of DL also came new architectures such as convolutional and pooling layers as novel network architectures to enable the usage of large number of input variables and recurrent neural networks for sequential data.

A standard neural network (NN) consists of many simple, connected processors called neurons, each producing a sequence of real-valued activations. Neurons be-

come active due to weighted connections from previously active neurons. According to the activation function some neurons may influence the environment by triggering actions. Learning is about finding weights that make the NN exhibit desired behaviour. Depending on the problem and on the model architecture, obtaining the desired output may require extensive and progressive iterations of the model. On each computational stage the aggregate function of the network is transformed [33]. Deep Learning is about accurately assigning values across such transformation stages.

2.2.5 Deep learning applications

Deep learning took off so quickly due to, manly, two reasons. First, it offered better performance on many problems when compared to traditional machine learning methodologies. Along with that, these architectures also make problem-solving much easier, because it completely automates one of the most crucial step on machine learning workflow: feature engineering. These two essential characteristics of deep learning made this approach so famous which has dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many others technological subjects.

On the subject of image recognition, since 2012 that deep convolutional neural network is the state-of-the art. This breakthrough carry out by Krizhevsky *et al.* [34] used a convolutional net to almost halve the error rate for object recognition and precipitated the rapid adoption of deep learning by the computer vision community. The neural network, which has 60 million parameters and 650,000 neurons was used to classify the 1.2 million high-resolution images in the *ImageNet LSVRC-2010* contest into the 1000 different classes. This work was also one of the first to successfully implement the *dropout* method to avoid overfitting.

On the topic of speech recognition, deep learning has seen its first major industrial application. On a joint paper from the major speech recognition laboratories [35] they all agree that deep neural networks, contain many hidden layers and trained with new methodologies have shown to outperform, until then, state-of-the-art methods on a variety of speech recognition benchmarks, and sometimes, by a large margin.

Also on the biomedical filed, deep learning has made some interesting contributions, although its adoption has been slow. As explained earlier, deep learning techniques are promising in extracting high level abstractions from the raw data of very large,

heterogeneous, high-dimensional data sets. This is exactly the type of data biology now has to offer [36]. For example on the topic of genomics, the implementation of next generation sequencing has contributed to the creation of massive amount of genomic data. Particularly on the topic of metagenomics there are several informatic challenges,being the major related to functional analysis of sequence data. Using deep learning architecture, specifically deep belief networks and recurrent neural networks, has allowed metagenomics classification and provided the ability to learn hierarchical representations of a data set [37].

Besides these refereed topics, deep learning has also made some interesting, and some time curious, improvements on the following fields [38, 36]:

1. Computer vision;
2. Restore colours on black and white images and videos;
3. Real-time pose estimation;
4. Describing photos;
5. Real-time analysis of behaviours;
6. Translation;
7. Computer games;
8. Self-driving cars;
9. Robotics;
10. Music composition;
11. Replications of painting styles;
12. Writing computer code;
13. Handwriting;
14. Deep Dreaming.

2.2.6 The rise of Deep Learning for Drug Discovery

In the past few years, DL has been attracting the attention of academic community and large pharmaceutical industries, as a viable alternative to aid in the discovery of new drugs [39].

Computational drug biology and biochemistry are broadly applied on almost every stage in drug discovery, development, and repurposing, as shown before. Worldwide many research groups have been developing different computational approaches for these fields and have successfully reduced time and resource consumption. Despite the implementations of these methods, none are yet optimal, having some limitation [36]. Furthermore, several studies now show that deep learning is an important approach to consider and explore.

Feature-based methods take their inputs in the form of feature vectors, representing a set of instances (i.e. drug-target pairs) along with their corresponding class labels (i.e. binary values indicating whether or not an interaction exists). Decision tree (DT), RF and support vector machines (SVM) are typical feature-based methods to build classification models based on the labeled feature vectors.

Deep architectures describe the data more precisely by encapsulating the most relevant higher-level abstractions. However, the efficient implementation of deep networks is still a major concern, since most of the current methods require solving a complex and non-convex optimization problem. On the other hand, extreme learning machines deal very efficiently with large-scale data, achieving fast learning speed and good performance, but employ a single hidden layer feedfoward network, not benefiting from the stacked generalization principle.

In fact, one of the first works in which DL was successfully applied was to solve problems related to QSAR on the machine-learning challenge posted by Merck. Surprisingly the proposed deep learning architecture improved Merck's in-house systems of approximately 14% on accuracy. QSAR models uses neural networks and is used in the pharmaceutical industry for predicting on-target and off-target activities. Such predictions help prioritize the experiments during the drug discovery process. Since 2012 QSAR models have been substantially improved, and inclusive have been enhanced with more recent deep learning techniques, such as deep neural networks [40]. On this work deep neural network (DNN) showed slightly better performance in 13 of the total 15 targets than the standard RF method. Along with this improvement the study also allowed to conclude about some characteristics of these architectures, such as [41]:

(i) DNNs can handle thousands of descriptors without the need of feature selection; (ii) dropout can avoid the notorious overfitting problem faced by a traditional ANN; (iii) hyper-parameter (number of layers, number of nodes per layer, type of activation functions, among others) optimization can maximize the performance of the algorithm;

Koutsoukas *et al.* [42], on his recent work, compared a DNN model with some commonly used machine learning methods such as SVM, RF, among others and proved that DNNs can outperform other machine learning methods.

One of the most important task on drug discovery, and the one under study on this work, is drug target interactions, which is still a major challenge in drug repositioning. Although there has been no explicit comparison with other machine learning methods the results indicated that DL can achieve a better performance.

Segler *et al.* [43] on his work on reaction prediction used 3.5 million reactions, taken from the collective published knowledge, as the training set for DNN, achieving a top-ten accuracy of 97%. The goal is to learn patterns in the molecules' functional groups that allow the machine to generalize to molecules it has not seen before. It is a multiclass problem, since there was previously named reaction that the network has to learn.

The first time that a deep learning method was employed to predict DTIs was last year by Wen *et al.* [20]. In this work an effective deep learning method – deep belief network (DBN) was developed and applied to accurately predict new DTIs between FDA approved drugs and targets. The drugs and targets data were extracted from the DrugBank database (`https://www.drugbank.ca/`) [44]. This resource, from 2008, is a unique bioinformatics and cheminformatics database, which gather detailed information about drugs and drug-target interaction. DrugBank database was been used by many researchers on a variety of works, which makes easy to compare results. The method was called DeepDTIs. The approach uses a DBN model to effectively abstract raw input vectors and accurately predict DTIs. The features used on this case, for drugs and targets were automatically extracted from simple chemical sub-structure and sequence order information. DeepDTIs was tested against more conventional methods such as random forest, Bernoulli Naïve Bayesian and decision tree. The hyper-parameters for the network were determined by grid-search and the AUC, accuracy, sensitivity and specificity of test set were 0.9158, 0.8588, 0.8227 and 0.8953, respectively. When compared to the other classification models, DeepDTIs performed better because the number of positive DTIs is much fewer than that of negative in drug target space. Since the purpose of the model was to predict the true positive DTIs, true positive ratio (TPR) is a more important evaluation metric among the four evaluation metrics. For five positively predicted drug target interaction, the authors did not find any experimental evidence, from other databases or literature, that could support the result. However these predictions still have potentiality to be true positive DTIs, which indicates that the model

is practically useful in predicting novel DTIs and in drug repositioning.

Drug target interactions are fundamental to renovate current drug discovery processes. Identifying the interaction between a drug and a specific biological entity will help to reduce the time and costs associated to it. Furthermore will contribute significantly to drug repositioning, which is a topic that have been targeted by many academic researchers and companies. The recent success demonstrates that deep learning methods are well suited for modelling complex biological data to support drug discovery and toxicological research. There is a need for more work on the area, with new approaches to keep on exploring this hot topic on a field that deserves many attention.

The applications of DL representations in the proteomics field are still few and recent and are still in the proof-of-concept stage with not many companies having been able to successfully apply these models for cost savings in drug discovery. Despite being successful when used, the full potential of deep architectures in the pharmacological field is yet to be shown, meaning there is still room for improvement. The availability of training data and collaborative efforts to collect data will be the game changer for deep learning on the topic of drug discovery.

3

Methods

Throughout the course of this work, we tried to develop a deep learning architecture do predict drug-target interactions. On this section, we will describe all the methods performed for the application of deep learning techniques, on Python 2.7, to the field of drug discovery. Topics from data collection and preparation to the evaluation metrics will be covered.

3.1 Data preparation

To perform the association study, data from different publicly sources has been gathered. In order to be able to easily compare the results with previously works on the area, very common DTI data bank were used. As previously stated, this work is based on the work carried out by Coelho *et al.* [19], and so the pipeline for data creation was the same. Again, we are facing a binary classification problem. We must predict if a specific pair of drug target, will or not interact, and there is a need to construct a dataset containing labeled data both for the positive and negative examples.

3.1.1 Positive data set construction

Known DTI data, meaning positive drug target interaction data, was collected from two different sources: (1) DrugBank [44] and (2) from a previous work on the field, a DTI prediction study by Yamanishi *et al.* [45].

The DrugBank database has been widely used to facilitate *in silico* drug target discovery, drug interaction prediction and other pharmaceutical fields. It provides detailed, up-to-date, curated, quantitative, analytic or molecular-scale information about drugs, drug targets and the biological or physiological consequences of drug

actions. First released in 2006, this database has been updated over the years, which makes it the centre of many studies. The version used was 4.3.

All DTIs were conveniently represented as a list of pairs. For each target the protein sequence was also extracted and the SMILE format for each drug. SMILE stands for simplified molecular-input line-entry systems, which is a line notation for entering and representing chemical structures and reactions. SMILES contains the same information as an extended connection table and is very compact relative to other methods of representing chemical structures. Any drug or protein for which the SMILE representation or protein sequence was missing or invalided, was just deleted.

From the SMILE representation, related to drugs, the chemical structure data and physicochemical descriptors were retrieved and encoded. From the primary protein sequence for the targets, a variety of physico-chemical descriptors were retrieved. All of these descriptors were used to generate our initial feature vectors that represent each DTI pair.

The second source of information, comprises DTI data from many different works, and has been used as a gold standard in several DTI studies. One database used on Yamanishi's work was DrugBank, so it was necessary to remove all the duplicated data between both databases.

On opposite to some works, here we disregarded the specific classes of protein targets. Some studies developed different models for each target type and assess it accordingly to that division. Here we considered the target as one single type, instead of divide into enzymes, G-protein, ion channels and nuclear receptors.

The number of unique drugs in our positive data sets was 2,118, being 1,328 from the DrugBank database and 790 from the Yamanishi's gathered data. Respectively to targets, the number is 2,077 (706 for DrugBank and 1,371 from Yamanishi). This makes a total of known positive DTI pairs of 10,736.

3.1.2 Negative data set construction

Ideally, the construction of negative data set should be, as well as the positive, based on known negative interaction between a drug and its target. However, there are very few authors publishing results for non-interacting DTI data, as these works are, naturally, associated with the failure of the hypothesis under study.

In order to overcome this issue, we considered drug target pairs with experimental bioactivity values greater than 10 µM, since for this range of values the pair is

considered to possess weak binding activity. So we screened the BindingDB [46] and BioLip [47] databases for drug target pairs that would fulfil this requirement. All the data was merged, duplicated were eliminated, resulting on 16,209 unique negative DTIs.

3.1.3 Final data set construction

In Figure 3.1 is shown a schema of the data set construction for this work.

By gathering data from different sources we could be making the mistakes of construct a data set without discriminative power. To test this, we used OpenBabel to extract the molecular fingerprints of the drugs in our data sets and to perform a comparison on their chemical similarity. With this analysis we could mislead cases of redundant information within our dataset.

OpenBabel is an open-source chemical toolbox that speaks the many languages of chemical data [48] which presents a solution to the proliferation of multiple chemical file formats.

In fact, less than 1% of all possible drug-target pairs had a sequence similarity score greater than 0.85, which means that the approach assumed so far might help to achieve the goal of developing a prediction mechanism. So, both data from positive and negative data set were combined to construct the final DTI space.

3.1.4 DTI descriptors

Descriptors for both drugs and targets were gathered using *PyDPI* [49]. *PyDPI* is a python package and a powerful open source for the extraction of features of complex molecular data. It computes 6 protein feature groups composed of 14 features that include 52 descriptor types and 9890 descriptors, 9 drug feature groups composed of 13 descriptor types that include 615 descriptors.

PyDPI allows to computing commonly used structural and physicochemical features of proteins and peptides from aminoacid sequences, molecular descriptors of drug molecules from their topology, and protein-protein and protein-ligand interaction descriptors. A more detailed explanation of these descriptors is given in the original paper for this package [49].

We finish the drug target space with a total of 26,945 entries, described by 755 features.

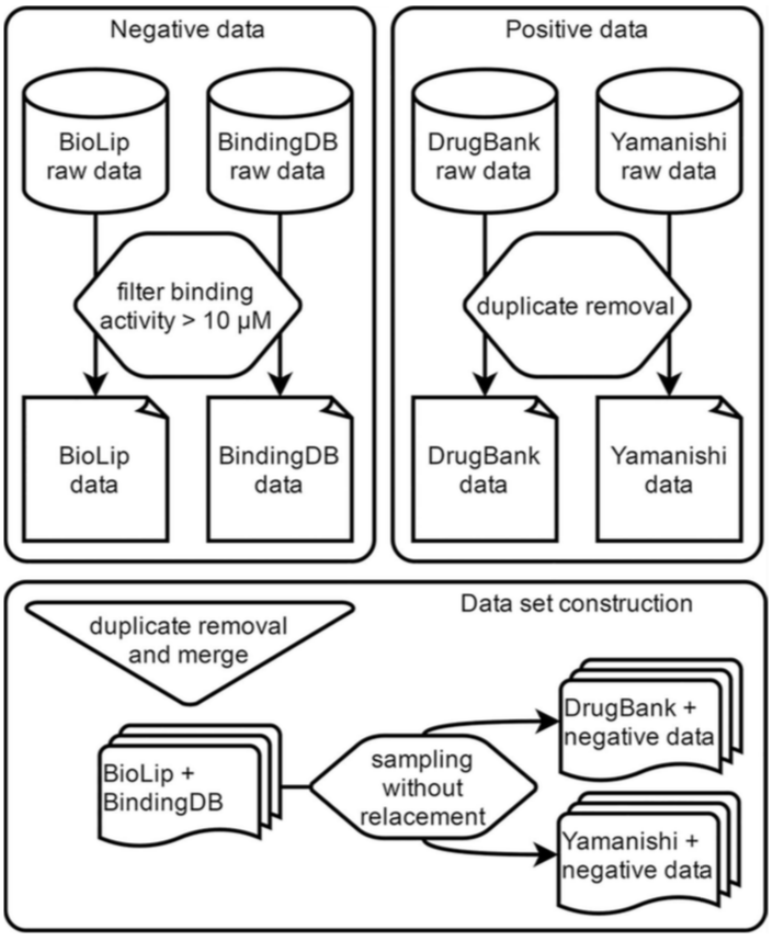

Figure 3.1: Data set construction based on the work by Coelho *et al.* [19].

3.2 Data analysis

Once gathered all the data, the next step was to analyse it. Some pre-processing techniques were applied to the data set before its splitting for further model training. In order to understand its behaviour and some possible correlation, some efforts were made in order to visualize the data, despite its high dimensionality.

3.2.1 Data exploration

Our data set is already on the form of floating-point data and was constructed in a way to avoid both missing and duplicate values, characteristics that saved many time on the data processing phase.

Even so, some hand-written functions were developed to ensure this preposition. We had to certify that no duplicates were present in the data set in order to avoid

redundancy, which could lead to major issues when classifying. On the course of this work we used *NumPy*, which is the fundamental package for scientific computing with Python, *Scikit-learn*, which is a free software machine learning library for the Python programming language and Pandas an open source library providing high-performance, easy-to-use data structures and data analysis tools. All these frameworks are essential for any machine learning exercise.

Analysing the dispersion of a specific feature (maximum and minimum values, uniques, among others), calculate balance ratio between both classes, compare two features space were some of the functions created to exploit the dataset.

3.2.2 Data pre-processing

Data exploitation allowed us to discover the differences between all features. In fact, some features had range values between 0 and 1000 when others were ranging from -1 to 1.

In general, feeding a deep learning architecture with a feature set containing either, large values and heterogeneous data will harm the model [32]. Doing so can trigger large gradient updates, which will difficult the convergence of the model. So, a common practice to deal with it is to do feature-wise normalization. To achieve this, for each feature we subtracted its mean to all values and divided by the standard deviation. This makes the features centered around 0 and with a unit standard deviation.

In order to perform the normalization we used *Scikit-learn* function *StandardScaler* which standardize features by removing the mean and scaling to unit variance. The normalization of the test set is done using values from the training data to keep test data completely unused.

Since the data set did not include missing values or duplicates, the normalization task was the only performed on the pre-processing step.

3.2.3 Data visualization

Our data set has a total of 26,945 entries, meaning drug-target pairs, and 755 descriptors. Being a numeric data set, and due to its high dimensionality, the best way to visualize all the data was applying a Principal Component Analysis (PCA)

so we could reduce the dimensionality in order to visualize it. PCA is a technique widely used for some applications, including dimensionality reduction.

By performing a PCA we will extract a set of new orthogonal variables of our dataset, called the Principal Components, that are able to represent the important information in a chosen number of vectors, called the eigenvalues. With PCA we are constructing new features, based on all the data, that we believe to have the best predictive power.

Since the goal was to be able to visualize the data, we applied PCA for computing two principal components. PCA could also be used for feature selection in some other cases. For more understanding on how PCA works please consult the work by Bro *et al.* [50].

3.2.4 Data splitting

Evaluating a model always boils down to splitting the available data into two different sets, train and external validation. The train set is used for training the model and the validation set to evaluate its performance. Once the model is ready for prime time, it is tested again with never seen data. So, this was the approach carried out on the course of this work.

3.2.4.1 K-fold cross validation

K-fold cross validation approach, could help to achieve even better performance for a model. It requires splitting the data into K partitions of equal size. Each partition is then divided into K parts again, and the model is trained with K-1 parts and assessed with 1 part. This process is repeated for all K divisions and the final score is the average of the K scores obtained.

In Figure 3.2 we can see an example of a four-fold validation, where the painted part represents the data subset used to evaluate the model performance, for each of the K partitions.

3.3 Feature engineering

Two of the most important aspects of machine learning models are feature extraction and feature engineering. Those features are what supply relevant information to

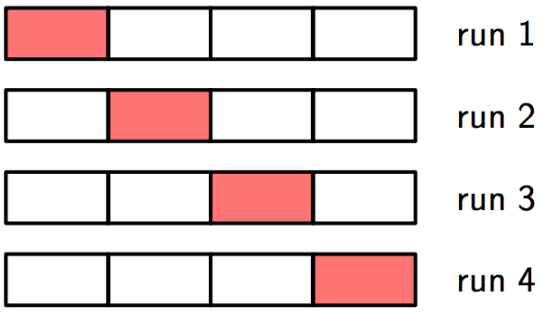

Figure 3.2: Scheme of k-fold cross validation [51]. The coloured sections represents the set of data used to evaluate the model.

the machine learning models. Actually, *Scikit-learn* already provides a function to directly select the k-best features for a specific model, which makes this task much easier.

Deep learning is changing feature engineering reality, according to its promoters. With deep learning, one can start with close to raw data, as features will be automatically created by the neural network as it learns. The ability to create gradually more abstract representations of data, makes feature engineering a less important step when developing deep learning architectures. In fact, some works on the area have proved that these kind of architectures can handle thousands of descriptors without compromising the model performance [41].

Before deep learning, feature engineering used to be a crucial step on machine learning tasks, because classical shallow algorithms did not have the ability to learn useful features by themselves, like deep learning does. Fortunately deep learning removes this need, because neural network can, without human intervention, understand which features have more predictive power, from the raw input data. This approach reduces drastically the need for a *a priori* engineering of more sophisticated descriptors of the data [52].

Based on these assumptions, and because on DTIs prediction problem there is not an understanding on what biological characteristics could help the most on the classification task, we decided to used the entire dataset as an input to our model. With this approach we can actually infer if, in fact, our architecture is capable to determine by itself what features are more important and achieve satisfied results.

3.4 Classification model implementation

Having the complex task of predicting whether a drug target pairs will interact or not, using the previously mentioned features, our main goal was to provide a classifier, based on a deep learning architecture, with a good perform and the ability to generalize to new data, never seen before.

Our approach was to first develop a baseline for the prediction problem and check its results. Then understand how we could improve the model, until achieving the best one used to predict on novel data.

3.4.1 Frameworks

In order to build, train and test a deep learning architecture we decided to use TensorFlow (TF) and Keras, both technologies compatible with Python.

TF is, nowadays, the most popular framework among many others, such as Torch, Caffe or Theano. We've found that TensorFlow, in particular, is easy to use, debug and monitor during and after training, and so it was our choice. This software library was developed by Google Brain Team for the purpose of conducting machine learning and deep neural network research. The name comes from the ability of working with tensors, which are multi-dimensional arrays.

Keras provides a high level Application Programming Interface (API) to build deep learning models. Keras can be used on top of other library, such as TensorFlow, and uses an object oriented design, resulting in a clean and user friendly interface. Furthermore, everything is well documented, which makes it one of the best tools to start with deep learning. As its creator François Chollet said: *'The library was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.'*

One of Keras' biggest advantages when compared to other APIs is its modularity. Keras turns a model into a sequence of standalone modules that can be combined together.

3.4.2 Deep feed forward network

There are many types of learning machines and many version of each. For this work we decided to used a deep feed-forward network. This network, is an example of a

deep learning model.

Since our data was numerical, and did not had a sequential correlation between the examples this seamed the best approach. Furthermore, this kind of architecture have proved to perform well on extracting, sequentially, abstract representations of raw data, which applies to the problem we are targeting. DNN are also used to supervised learning classification problems, which is our case.

The goal of a feed-forward network is to define a non linear function, capable of mapping the input data into the desired output, as

$$y = f(x : \theta)$$

and to learn the value of all parameters θ belonging to the network, in order to achieve the best function approximation [25]. These networks are typically represented by composing together many different functions.

The network is composed of several layers, and each layer of many neurons. A neuron combines input from the data with a set of coefficients, called weights, that either amplify or dampen that input, thereby assigning significance to inputs for the task the algorithm is trying to learn. These input-weight products are summed and then pass through a activation function which determines whether and to what extent that signal progresses further through the network to affect the ultimate outcome. This is how the information is passed between layers. On Equation 3.1 is represented the output of a neuron, called activation, which depends, as explained, on the activations of the neurons on the previously layer. Activations of one layer, determines the activations on the next layer.

$$a_j = \sum_{i=1}^{N} x_i w_{ij} + w_{j0} \tag{3.1}$$

where N represents the number of neurons on the previous layer *j-1*. We shall refer to the parameters w_{ij} as weights and the parameters w_{j0} as biases. The quantities a_j are the activations. Each of them is then transformed using a differentiable, non-linear activation function, before being used as input for the next layer. Activation functions are responsible for enhance the model to achieve non-linear functions. Rectified linear unit (ReLU) is the most popular activation function in deep learning, although there are much more such as Sigmoid (Sigm)or Tanh. On 3.3 we can see an resume of all this concepts.

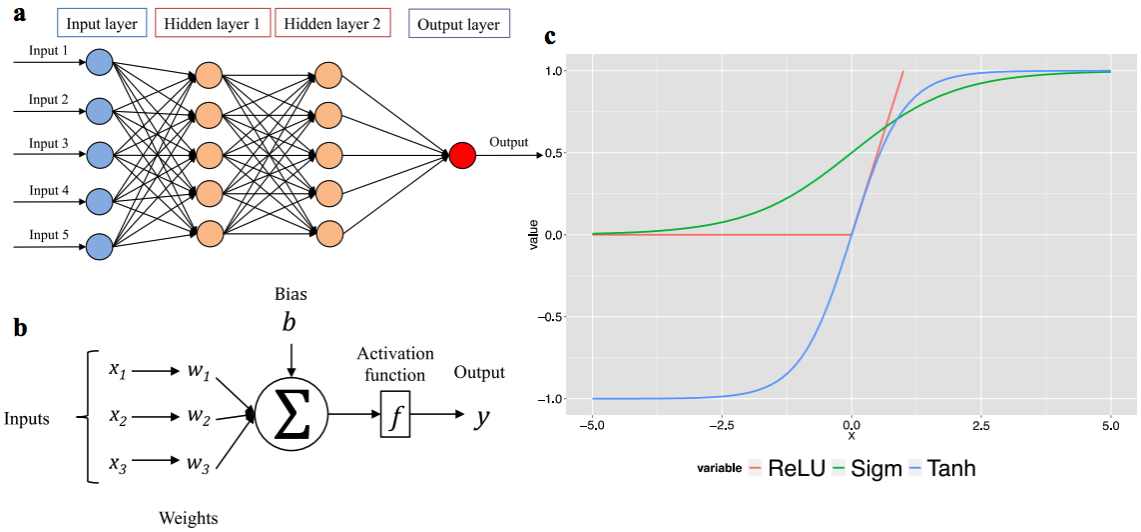

Figure 3.3: **a** Example of a feed forward network with two hidden layers and multiple neurons. **b** A schema of the mathematical formulation for each neuron of a DNN. **c** Popular neurons activations functions [42].

In order to achieve the desired output, we need to measure how far, from it, is the model performing. This control is established by the loss function, defined by the following mathematical formulation. We'll call *E(w)* the quadratic cost function; it's also sometimes known as the mean squared error or just MSE.

$$E(w) = \frac{1}{2n} \sum_{x} ||y(x) - a||^2 \tag{3.2}$$

Here, w denotes the collection of all weights and biases of the network, n is the total number of training inputs, a is the vector of outputs from the network when x is input, and the sum is over all training inputs, x. This is the typically used loss function for linear regression models. When related to deep architectures, this loss function must be chosen accordingly to the problem to be solved.

In summary, there is a natural choice of both output unit activation function and matching error function, according to the type of problem. On our case, for binary classifications, logistic sigmoid outputs and a cross-entropy error function are the most common choices.

3.4.3 Parameter optimization

We turn next to the task of finding a weight and biases vector w which minimizes the chosen function *E(w)*. At this point, it is useful to have a geometrical picture

of the error function, which we can view as a surface sitting over weight space as shown in Figure 3.4.

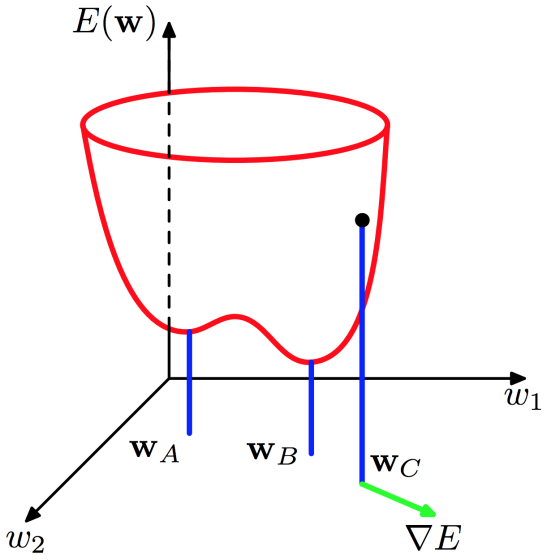

Figure 3.4: Geometrical view of the error function *E(w)* as a surface sitting over weight and biases space. We can see a local minimum and the global minimum, which is the point where the cost function will be lower, and so the model performance the best. At any point w_C, the local gradient of the error surface is given by the vector ∇E [51].

In order to reduce the loss function output, we used the so called gradient descent algorithm, which iteratively adjusts parameters, gradually finding the best combination of weights and bias to minimize loss. Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. A general function *E(w)*, may be a complicated function of many variables, not like the one in Figure 3.4, and it won't usually be possible to just eyeball the graph to find the minimum. There is a need for an algorithm to gradually achieve this minimum.

If we make a small step in the weight and bias space from $\boldsymbol{w}$ to $\boldsymbol{w} + \delta\boldsymbol{w}$, then the change on the error function is $\delta \mathrm{E} \simeq \delta\ w^T\ \nabla E(\boldsymbol{w})$. The gradient vector point into the direction of greatest rate of increase of the error function. Since the error function is a continuous function of the weight and bias space, its smallest value, and our goal, will occur at a point that the gradient of the error function is null, so that:

$$\nabla E(w) = 0 \tag{3.3}$$

While this is not achieved the model should make a small step in the direction of $-\nabla E(\boldsymbol{w})$, which is the direction where the error function decreases quicker. The parameter related to this small step is the learning rate. The learning rate is introduced as a constant, normally step as a very small value, which force the weights to get updated, accordingly to the error, in a very smoothly and slowly way, in order to avoid big steps and chaotic behaviour. After each such update, the gradient is re-evaluated for the new weight vector and the process repeated. High learning rate means faster learning, but with higher chance of instability. Also to define the correct value for learning rate a study is necessary. Along with learning rate we have momentum, which controls how much to let the previous update influence the current weight update.

There are three variants of gradient descent, which differ in how much data we use to compute the gradient of the objective function. Depending on the amount of data, we make a trade-off between the accuracy of the parameter update and the time it takes to perform an update. Batch gradient descent or vanilla gradient descent, stochastic gradient descent (SGD) and mini-batch gradient descent are the three most used variants of gradient descent.

Backpropagation algorithm, which is a fast way of computing the gradient of the cost function, allow to pass the partial derivative of the error with respect to each parameter in a backward pass through the network. This algorithm, also known as backprop, allows the information from the cost functions to flow backward through the network, in order to compute de gradient and update the parameters. Actually, back-propagation is nothing else, than the method for computing the gradient.

The update performed on the weight and bias space depends on several methods called optimisers Adagrad, Adam, RMSprop [53]. The delta rule is using the most simple and intuitive one, however it has several draw-backs.

Furthermore, the batch size, which is the number of examples, or the number of patterns, used in one iteration, that is, one gradient update of model training, also needs to be assigned. For example, the batch size of SGD is 1, while the batch size of a mini-batch is usually between 10 and 1000. Batch size is usually fixed during training and inference.

The full data set can also be shown to model, during the training process, several times. The number of epochs is, precisely, the number of times that the entire training data set is presented to the model.

In Figure 3.5 we can see a schema of all the parameters that were referred before

and that together, explains how a deep neural network perform.

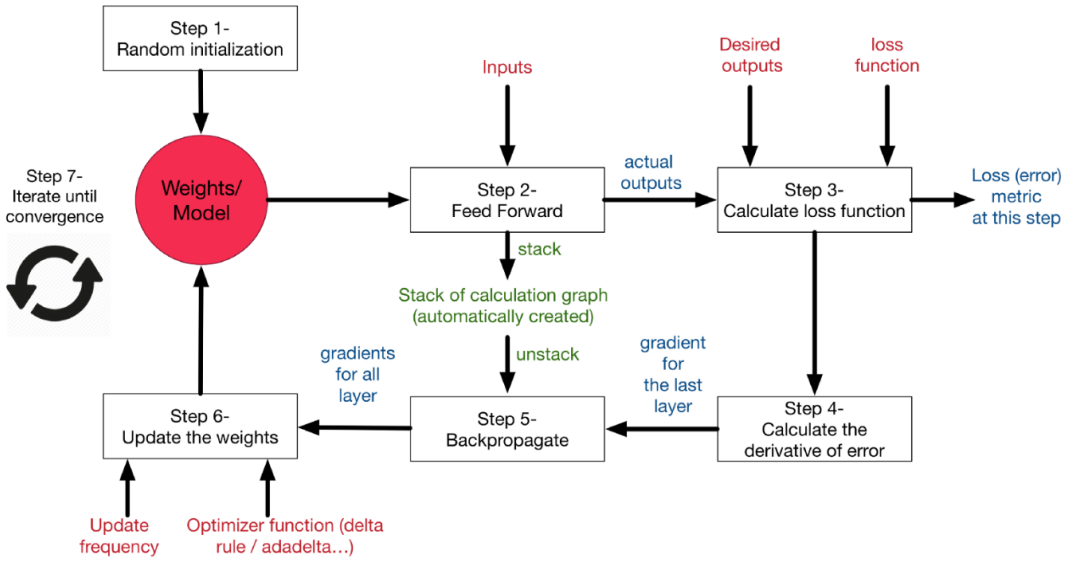

Figure 3.5: Step-by-step of how a neural networks performs and all its constituent parameters [54].

We end up with many parameters, from number of layers and neurons, to activations functions and learning rates, which all together complete the network we are developing. All of these parameters were integrated on our work, in order to achieve a complete and empowered model. As already stated, correctly assign these parameters is a challenging task with great influence on the model's performance. On this work, and in order to achieve the best set of parameters we performed a hyper-parameter tuning.

3.4.4 Hyper-parameter tuning

One main aspect of deep neural network are their main hyperparameters. These can be either architectural, such as the number of layers and neurons, the activation function among others, optimization types, such as the learning rates or regularization such as the dropout probabilities. Hyper-parameter tuning is the process of finding the optimal combination of those parameters that minimize the cost function.

The reason is that neural networks are notoriously difficult to configure and there are a lot of parameters that need to be set. On top of that, individual models can be very slow to train. Optimization all these aspects on such architectures is, in general, extremely challenging due to the large number os parameters to evaluate.

This is definitely the most difficult and time consuming task when building a deep learning model.

One way of achieve this is trial and error method, which was our first approach. Quickly we realise how hard and time-consuming it would be, and so, the strategy was changed.

3.4.4.1 Grid-search

The best way to find the best set of parameter for our model was through grid-search method, and Scikit-learn has a proper function to do so. This strategy consists on creating a grid on the space of possible parameters and systematically check for each grid vertex what the value assumed by the cost functions is. In other others, parameter are divided into buckets, and many different combinations are tested. In the end, we can understand which set of parameters best fit our model. In order to do a search, we need to define a hyper parameter space, that is, all the hyper parameters we want to test and their possible values.

In this work we performed a grid-search for hyper parameter tuning on many the parameters involved on the model construction. Due to the time that the grid-search required to run, the parameters had to be split into groups. These groups were constructed taking into account that some parameters might be connected and influence together the model. For example, there is a dependency between the amount of learning per batch (learning rate), the number of updates per epoch (batch size) and the number of epochs, and so, the grid-search to compute the best set of these parameters should perform at once.

For every grid-search a different script was written. The choice of the best value for each parameter was achieved comparing the results of accuracy for 10-fold cross validation, between all possible combination of parameters. All the following parameters were define based on a grid-search, where the possible range for them was constructed based on theory and some manual tuning done previously.

1. Batch size;
2. Number of epochs;
3. Training optimization algorithm;
4. Learning Rate;
5. Momentum;

6. Weight Initialization;
7. Neuron activation function;
8. Number of neurons;
9. Number of hidden layers;
10. Loss function.

In the end, we get the set of parameters that best fit the model, in order to achieve the lower cost function, and consequently achieve better results. With these parameters our deep feed forward network was concluded and ready to be tested.

3.4.5 Model improvement

Besides all the tuning methods we also performed some strategies to improve model performance and avoid overfitting. Between many methods for model control, we decided to use dropout method, recently proposed by Geoff Hinton's group.

The methodology used was dropout regularization, which is commonly used on neural networks. Dropout regularization works by removing a random selection of a fixed number of the units in a network layer for a single gradient step. The more units dropped out, the stronger the regularization is. This is analogous to training the network to emulate an exponentially large ensemble of smaller networks. This prevents units from co-adapting too much, during training, and has been shown to improve model performances [55] also on computational biology. One of the drawbacks of dropout is that it increases training time. A dropout network typically takes 2-3 times longer to train than a standard neural network of the same architecture.

Along with dropout methods, and since we are facing a binary classification problem, we also investigate what should be the appropriate threshold for consider an example as positive. Some works on the area have imposed this threshold, instead of 0.5 as normal, to much higher values, such as 0.9. Changing this will make much less pairs of drug target to be classified as positive.

3.5 Evaluation metrics

Four of the most frequently used evaluation metrics are: area under the receiver operator characteristic curve (AUC), accuracy (ACC), true positive ratio (TPR)

which is also known as sensitivity or recall and true negative rate (TNR), known as specificity. All these metrics were used to assess our model performance, since we are towards a binary classification problem. The calculation of the formulas of ACC, TPR and TNR are the following:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.4}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.5}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3.6}$$

where TP, FP, TN and FN are true positive, false positive, true negative and false negative, respectively. In our binary classification problem, the output is labeled as positive or negative. If the prediction and actual value are all positive, it is called TP, otherwise if the prediction value is positive while the actual value is negative it is called a FP. The same for when the prediction is negative. With this values we also computed a confusion matrix, which, itself is relatively simple to understand and summarize all the results.

Besides this four metrics, we also include F_1 score known as balanced F-score or F-measure as an evaluation metric. With F-score we can considerer both precision and recall. While recall expresses the ability to find all relevant instances in a dataset, precision expresses the proportion of the data points our model says to be relevant and actually are.

Recall is calculated through Equation 3.5. Precision is defined by the following formulation:

$$Precision = \frac{TP}{TP + FP} \tag{3.7}$$

F_1 score is the harmonic mean of precision and recall taking both metrics into account in the following equation:

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{3.8}$$

We use the harmonic mean instead of a simple average because it punishes extreme

values. For unbalanced data, this F_1 score is usually considered to be a better criterion to assess the prediction performance, since it can punish more false positive examples.

We also computed a confusion matrix, which is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand and uses TP, FN, TN and FN.

4

Results and Discussion

4.1 Drug target space treatment

Using PyDPI we calculated 755 descriptors for each DTI. 432 of these descriptors where related to proteins and 323 to drugs.

The target descriptors are shown in Table 4.1 and drug descriptors in Table 4.2.

Type of descriptors	**Number**
Aminoacid composition	20
Moran autocorrelation	240
Physicochemical	141

Table 4.1: Different types of descriptors for targets.

Type of descriptors	**Number**
Molecular constitutional	30
Molecular connectivity	23
Molecular property	6
Shape attributes	7
Electronic (charge)	12
Molecular Access System	166
E-state fingerprints	79

Table 4.2: Different types of descriptors for drugs.

We end up with a drug target space of 26,945 pairs, described through 755 features. Example of non-interacting pairs represented 60% of the data set, which approximates to a balanced data, and so facilitating the classification problem.

In order to visualize the data set, we applied PCA to reduce it to a 2D problem, so we could plot the points accordingly to its label. In Figure 4.1 we can see the result.

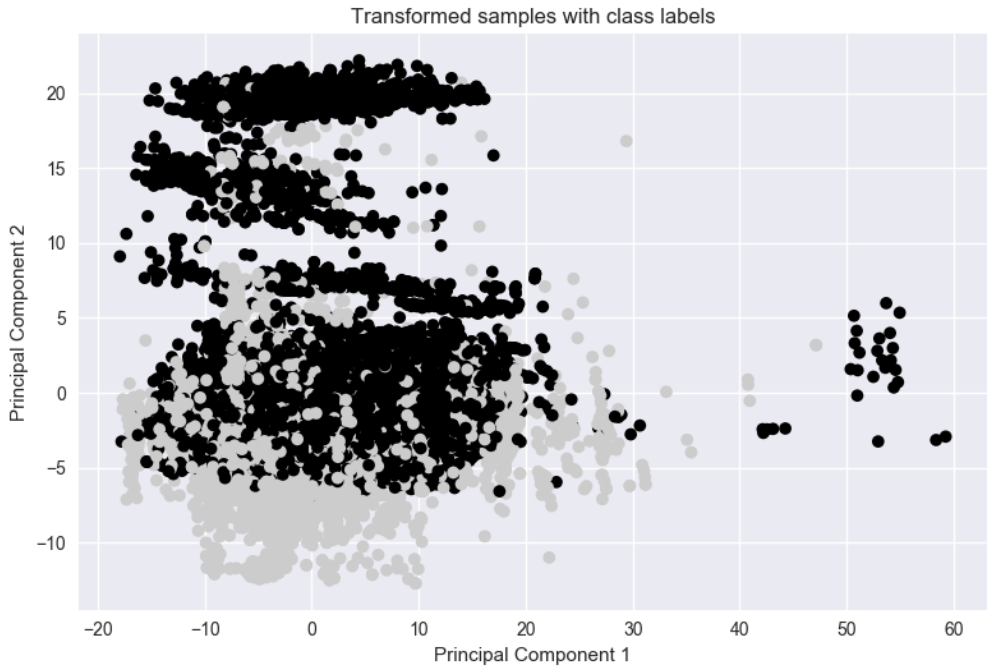

Figure 4.1: 2D visualization of the data set, after applying PCA.

Apart from the points where first principal component is high, the spectrum of points seams to overcome. We quickly realise that many points are together and even overlapping. Its is impossible to identify locations for each label or any kind of cluster. This result proves that DTI prediction is not a linear problem and that some traditional classification methods may fail, reinforcing the need for studying deep learning techniques on this field, due to its ability of learn abstracts representations of the data.

This non-linearity of the data was also verified when analysing the dispersion of some features among the classes. In Table 4.3 we can see how a specific binary feature behaves within the different classes and see that there is not a clear division of the data. Like this feature, many other had a similar distribution.

Since we had more examples for the negative instances, each instance of this data set was randomly selected and append to one of the positive data sets, beginning with *Yamanishi* and then DrugBank, until all instances were attributed. With this

	0	1
Positive	1198	9714
Negative	1170	6036

Table 4.3: Dispersion of feature 700 between the two classes.

procedure we kept the positive negative ratio. This led us to a training data set of 18,118 instances, being 7,206 from the *Yamanishi* data and 8,827 pairs for external validation.

4.2 Baseline approach

The first goal for building the deep learning model was to define a baseline. This baseline would serve as a reference point for comparing how well our model is performing. A baseline helps to quantify the minimal, expected performance on a particular problem. Our baseline is a naive and simple approach to DTI prediction problem. In Figure 4.2 is shown the structure of our baseline model.

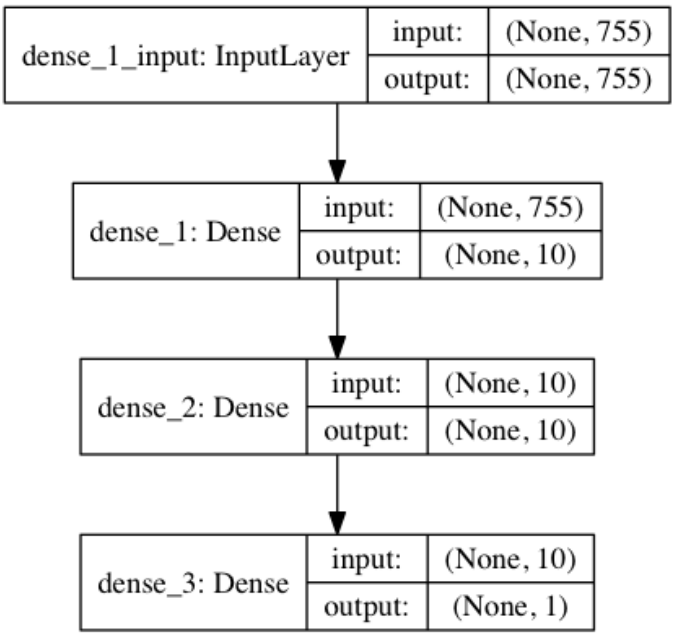

Figure 4.2: Baseline architecture.

We used 4 fully connected layers, called dense layers. The input layer, naturally had 755 neurons, one for each of the features, both hidden layers had 10 neurons and

the output had only one, because we want the output to be a probability. Input and intermediate layers use ReLU as their activation function and the final layer use a sigmoid activation in order to output a probability between 0 and 1. The loss function chosen was *binary_crossentropy*, since we are facing a binary classification problem, although its not the only viable choice. The model was trained using Adam was the optimizer, 20 as the number of epochs and batches of 1000 samples.

Surprisingly our baseline achieved an AUC of 0.86 on the external data, which is close to the results of other works on the area. In fact, such a simple approach almost evened the results of other authors, which made as thinks that, by improving the baseline could lead to excellent results.

4.3 Architecture decisions

Model improvement can be achieved by changing its architecture. There are two key architecture decisions to be made for a stack of dense layers. Firstly the number of layers to use and then the number of hidden units, or neurons, to include in each layer. So we start with these two parameters.

Having more hidden units allows the network to learn more complex representations of the data, but it makes the network more computationally expensive and may lead to learning unwanted patterns. Such patterns may increase the performance on training data but not on the test data.

We did not find an efficient method that could help on the discover of the best number of layers and neurons for each layer at once. Our first approach was to manually try deeper networks, with many layers, and wider networks, with many neurons per layer. We soon realize that the best performance was achieved with a balance of this two characteristics. In fact, by creating a deeper or wider networks the improvements of model performance were minimal and a complex model is more conducive to overfitting.

We managed to separately use grid search, described on 3.4.4.1, for both number of layers and neurons. On this method, we define a set of possible values for these parameters and test all possible matches. In order to extract meaningful conclusions from these methods, we used 10-fold cross validation when assessing all the possible combinations.

Since we had no clue about the correct value for each parameter, the range of

possibilities was wide, so it could include both smaller and higher values. We used the following strategy to determinate number of layers and neurons parameters:

1. Grid-search for the number of neurons for input layer;
2. Grid-search for the most suitable number of hidden layers;
3. Grid-search the remaining numbers of neurons for each layer.

For the first step, the result of the grid-search showed that input layer should have 25 neurons, achieving an accuracy of 0.956. This seemed a small number, when compared to the input size, which is composed by 755 features. This discrepancy might indicate that many features do not have prediction power. In fact, increasing the number of neuron for the input layers affected negatively the model performance.

After defining the input layer, with 25 neurons, and the output layer, with, naturally, one neuron, we start the search for the best number of hidden layers. Once again, grid-search showed us that 3 hidden layers would be enough to construct our model and get an accuracy of 0.974. This makes a total of 5 layers.

The next step was to determinate the most suitable number of neurons for these 3 hidden layers. A sequence of grid-search showed that the number of neurons for all the 3 hidden layers should be: 50, 100 and 10, achieving an accuracy for each of 0.974, 0.975, 0.976, respectively. On 1^{st} and 2^{nd} hidden layers the number of neurons increases, what might suggest that the model is learning abstract representation of the data, and on 3^{rd} the number decreases before passing the information to the output layer, which has only one neuron. The accuracy of the model, based on 10-fold cross validation increases gradually, step by step.

So, we implement a model with such characteristics, and continue to analysing other parameters, always considering these architecture as a base.

When it comes to decide the best number of epochs, we plot the accuracy behaviour, on the training set, accordingly to the number of epochs, so we could understand the best value for this parameter. This is possible due the Keras callback function, which keeps a history of the model training. In Figure 4.3 we can see the result.

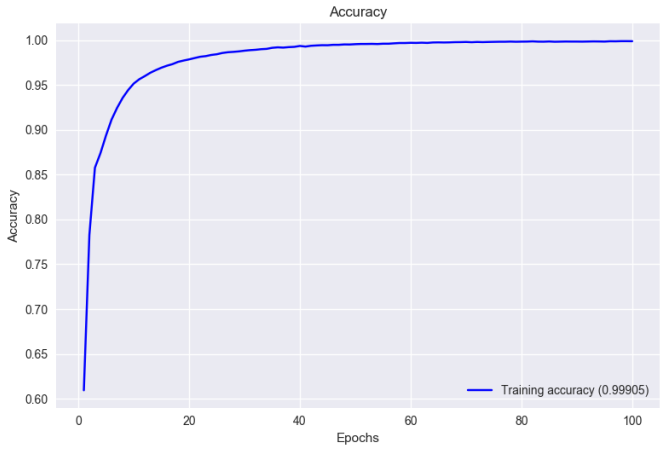

Figure 4.3: Model performance according to the number of epochs.

We decided to move forward with 50 as the number os epochs, since the accuracy performance stabilizes around this value. A higher number of epochs could lead to overfitting.

For the batch size and optimizer we decided to use, once again, grid-search method. Along with batch size and optimizer, we decided to include the number of epochs, since these parameters are related, and might influence each other. In Table 4.4 we can se the set of values chosen for each parameter.

Parameter	**Range of possible values**
Number of epochs	[30, 40, 50]
Batch-size	[50, 100, 250, 500, 1000]
Optimizer	['adam', 'SGD, 'rmsprop']

Table 4.4: Set of possible values for number of epochs, batch-size and optimizer used on the grid-search.

The results for the grid-search are summarized in Table 4.5. This set of parameters achieved a mean accuracy for 10 fold cross-validation of 0.976. As we can see the grid-search output for number of epochs was the same as we previously defined in Figure 4.3, which validate both approaches.

With the optimizer Adam chosen, we move forward into understand be best learning rate and momentum for it. Grid-search showed that the best learning rate would be 0.001 and momentum 0.4. With all the parameters defined so far, our model presents

Parameter	Grid-search output
Number of epochs	50
Batch-size	50
Optimizer	'adam'

Table 4.5: Grid-search results for number of epochs, batch-size and optimizer.

a 0.976 accuracy for 10-fold cross validation. Since the performance is already very high, the influence of the parameters is almost null.

Weight initialization was the last parameters defined through grid-search. As expected the accuracy for ten-fold cross validation did not improve, but the initialization with better result was lecun-uniform which is a method that can avoid main of the backpropagation limitations.

The activation function for the output layers was kept as a sigmoid function, and the remaining layers as ReLU, which is less prompt to errors, and proved to be the best by our grid-search method. The loss function also remained the same as our baseline, binary entropy, because is widely recognized for binary classification, this is the best function to use on backpropagation algorithm.

The grid search for parameters were computed in groups constructed taking into account that some parameters might be connected and influence together the model. These procedure might not be most correct one since, all parameter should be grid searched all together. Is the entire set of parameters that completes the model, and splitting them could lead to miss the best combination of all parameters. We are inferring the results of one grid search to the other.

Despite the efforts of performing a single grid search, taking into account all the parameters, referred below, computational power was not sufficient.

With all the parameter correctly assigned we finish the construction of our architecture. On Table 4.6 all the parameters used are summarized.

Besides all the parameters referred above, which are defined, and keep the same throughout the training process, our model is also composed of many other parameters, that are trained on the process. On Table 4.7 we can see all these trainable parameters and its distribution between layers. Among all this 26,321 parameters are the weights of each connections between neurons and its biases.

Parameter	Value used
Number of Layers	5
Total number of neurons	186
Batch-size	50
Number of epochs	50
Optimizer	adam
Learning Rate	0.001
Momentum	0.4
Weight initialization	'lecun_uniform'
Activation function	ReLU
Loss function	'binary_entropy'

Table 4.6: Final parameters used on the model constructed.

Layer type	Output shape	Trainable parameters #
Dense	25	18900
Dense	50	1300
Dense	100	5100
Dense	10	1010
Dense	1	11

Table 4.7: Trainable parameters (weights and biases) for the model.

With many tuning, and adapting all the parameters to the data, the model could be overfitting. Being all the parameters tuned for the training set, despite of using 10-fold cross validation, model could be memorizing the data shown, instead of learning new representations for it. In order to ensure that this was not the case, and that our model would performance well on new data we applied dropout method.

4.4 Network regularization

Drop-out rate allows model regularization in an effort to limit overfitting and improve its ability to generalize. To get good results, dropout was best combined with

a weight constraint such as the maximum norm constraint. This weight constrain function allows setting constraints, for example non-negativity on network parameters during optimization. We applied grid-search for dropout percentages between 0.0 and 0.9, 1.0 does not make sense, and maximum norm weight constraint values between 0 and 5.

The penalties are applied on a per-layer basis. Although, grid-search showed that dropout would improve the model if applied only on the first two layers. And so we did. We add dropout regularization to the input layer with a rate of 0.2 and a maximum norm weight constraint of 4, and for the second layer 0.1 and 4. In fact, we could improve 10 fold cross-validation to 0.979.

We are facing a binary classification problem so we decided to inspect the threshold used as a criterion that is applied to a model's predicted score in order to separate the positive class from the negative class. Normal binary classification would use 0.5 to make this division, although on our case this threshold was worth to explore.

Since our dataset has 60% of negative examples, by increasing the threshold, we would expect to get better results, because it would be harder to label a example as positive. In fact, after some manual search for the best, we conclude that a threshold of 0.8 would improve the model.

In Figure 4.4 we can see a structure of the entire pipeline, from data gathering to the final model capable of predict the interaction between a drug and a target.

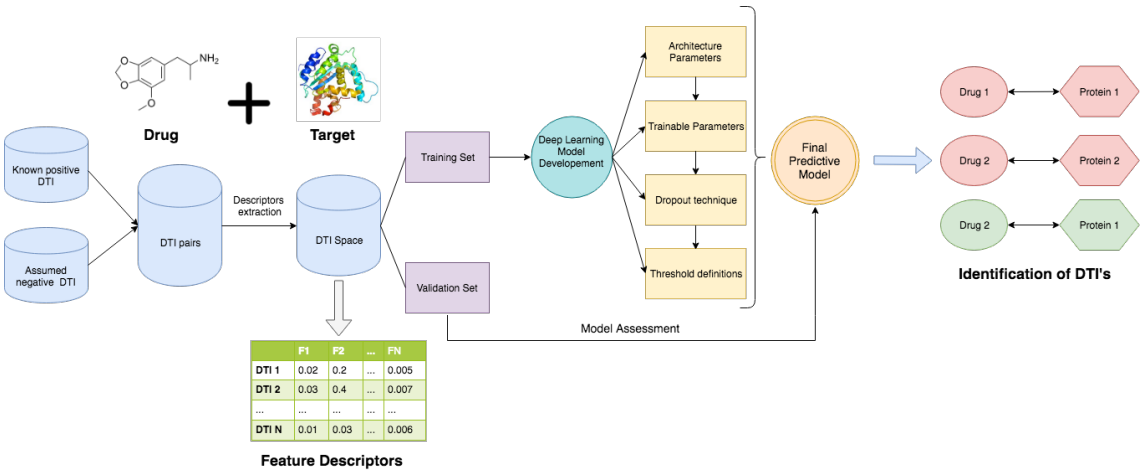

Figure 4.4: The flowchart of the proposed deep learning pipeline..

4.5 Evaluation metrics

In order to fully assess the performance of the model we calculated many evaluation metrics. So we could get more robust results, we run the model 100 times and calculated accuracy, recall, specificity, AUC and F1 for each run. In Table 4.8 we represent the mean and standard deviation for each of the evaluation metrics. All the metric, were calculated after using the model to predict novel DTIs pairs, never used before.

Evaluation metric	**Mean**	**Standard deviation**
Accuracy	0.904	0.003
Sensitivity	0.771	0.008
Specificity	0.990	0.002
Area under the ROC curve	0.880	0.004
F1-score	0.863	0.005

Table 4.8: Evaluation metrics results for 100 runs.

We also computed a confusion matrix to have a better visualization of the model classification. In Table 4.9 we can see an example of a confusion matrix for one run.

Classification accuracy is the number of correct predictions made as a ratio of all predictions made. In our case, the dataset is almost balanced so this metric can, actually infer about model's performance and give us a general overview. Achieving 90% of accuracy is a great insight that our model satisfies the goal of correctly predict if a drug and a target will interact. The best comparison, is the work by Wen *et al.*, referred on section 2.2.6, in which other DL architecture was applied to the same database, achieving an accuracy of 85,9%. The proposed model managed to achieve a higher accuracy, and so prove its value.

		Predicted label	
		0	**1**
True label	**0**	5130	65
	1	725	2609

Table 4.9: Confusion matrix for one run.

Recall or sensitivity is, in fact, the lowest evaluation metric for our model. Ideally we would want to maximize both sensitivity and specificity. Although there is always a trade-off. In our case we have a specificity of 99%. We decided to do so, in order to develop a method capable of avoiding false alarms. Since the main goal of the proposed architecture was to mislead candidates drugs that later will fail, the percentage of false positives must be lower.

AUC is a commonly used for binary classification methods by comparing true and false positive rates which shows the ability to discriminate between positive and negative classes. The achieved result is great, although some other traditional methods, applied to DTI also achieved this range of values. Probably experimenting other DL techniques, or exploring better and different way to tune the model, could lead to better results and, consequently outperform classic approaches.

The F-measure can be interpreted as a weighted harmonic mean of the precision and recall. With a value of 86% is also, demonstrative of the prediction power of our model.

The results obtained suggest that the proposed deep learning architecture can be used in the identification of new leads for drug repositioning, and should be used to improve drug discovery process. The results also prove that DL techniques could, as in many other fields, easily become state-of-the-art for drug target interactions prediction.

4.6 Final remarks

Deep learning is now starting its implementation on drug discovery field, and many works on this area will soon show up. Here we proved that DL techniques can outperform traditional methods when predicting the interaction of a drug and its target, like happens in other areas. We are also confident that our pipeline could be implemented on a drug repositioning flow, causing a reduction of the time and efforts usually associated with it. On a practical usage of our model we could screen through many drugs for different targets, prediction their interactions and identify putative candidates for drug repositioning.

Concerning the work developed here, more data could be collected. The quality of DL models is generally constrained by the quality of our training data. Furthermore DL techniques get better with more data, so the more data the best.

The construction of our baseline was an important step for the remaining work. First of all, it was our first contact with the frameworks used and helped us to dive into DL world. Secondly the quality of the results achieved with a simple approach encouraged to keep on with the study of applying deep architectures to drug discovery.

Our assumption on not to perform feature selection, due to the robustness of DL techniques to unrelated data, might have influenced the performance. The true is, network will use a near zero weight and sideline the contribution of non-predictive attributes. In fact, we saw that the optimal number of neurons for the input layer was 25, when the input consists on 755 features. This could suggest that, after all, many features are useless for the problem. Despite this ability of outwit unimportant features, there is always weights and training cycles used on data not needed. It could be interesting to perform some feature selection, based on some analytical tests, reduce the drug target space and understand if the performance of the model increases.

The hyper-parameter tuning was a really demanding task. As a matter of fact, constructing the model, with the all the tools provided, is relatively simple. The hard task is on tuning the model for all the parameters. A big part of the time dedicated to the practical component of this work was spent on all the tuning process. There are more than ten parameters to evaluate, understand how they affect the network and if they are correlated. Recent Python packages facilitate this process such as Hyperas or Deepreplay which allows even to visualize the tuning process. It was not possible to integrate any of this tools on the work pipeline, which made tuning a more iterative and complex process. Despite the efforts to do so, not making a single grid-search for all the parameter might also compromised the final results, due to the inner connection parameters might have. Still we tried to group parameters that we know to be correlated, such as the number of layers with the number of neurons, the number of epochs with the batch-size and so on. Another approach for model tuning could be studied and maybe lead to better results.

Overfitting is a very common problem when training a deep network. Here, in order to work this issue around, we decided to apply dropout methods. Although using dropout method was proved to improve out model other techniques for control the model could have been developed, such as early stopping or weight decay. Dropout method have proved to improve model performance and is the most used method to avoid overfitting, although alternative approaches could have been tested.

The accuracy achieved, which provides an overall evaluation of the model was, in

fact, quite good and better when compared to other works on the area. Due to the threshold changes, specificity values were near 100%, which results in a model capable of avoiding false alarms. This means that, when our model predicts an interaction between a drug and a target as positive, there is a high probability of being, actually, positive. Using our classification model on a practical flow for drug discovery, would decrease the rate of drug failures on the clinical trials, and consequently the resource and financial efforts behind the entire process. Despite the great results achieved, we believe other deep architectures could even perform better

On general DTI prediction problem there is also, many room for improvements. The field is lacking on methodologies applying other DL architectures such as, convolutional or recurrent neural networks. We still do not know how other DL architectures will behave on DTI challenge. Furthermore, drug discovery could benefit largely from the ability of DL to perform on unsupervised environments. In fact, deep learning can be widely unsupervised once set in motion and even learn intricate patterns from high-dimensional raw data with little guidance.

Who knows if these other architectures could, instead of only predicting if a drug target pair will interact, understand what kind of interaction would most probably happen, or properties of such interaction, like epoxidation. We could turn the problem into a multiclass one and definitely a more challenging one.

Traditionally, the performance of DTI prediction depends heavily on the descriptors used to represent the drugs and the target proteins, which is contradictory with the fact of not existing an agreement on which characteristics of a target and a drug could make them interact, and once again deep learning could help with that. Using deep learning as a feature selection method changes the actual paradigm. DL can understand which are the features that describe the data the best, or even construct new representation for it, with a great prediction power.

Not everything is good, the fact that deep learning represent data into gradually more abstract ways, it is many times called black-box. This name is given because we can not understand what the algorithm itself is doing, how it is transforming the data and which feature are important, ending up losing the ability to extract meaningful biological conclusions. This drawback has been the main reason for the delayed adoption of deep learning on biomedical topics.

To sum, over the past three to four decades, the use of computational methods in drug discovery settings has steadily increased and computations have become

an integral part of discovery research. Although drugs are not discovered and developed solely through *in silico* experiences, and predictions can not alleviate the need for experimental work, computational approaches make valuable contributions to the highly complex discovery process at different levels. Hence, understanding opportunities of popular *in silico* approaches, such as deep learning should be of considerable interest to a wide drug discovery and development audience, and this work contributes as a support for the interest that already exists and for what will come.

5

Conclusion

In 2016 and 2017 Kaggle was dominated by two approaches: gradient boosting machines and deep learning. Although, deep learning adoption on biomedicine has been slow and so this works intends to contradict this resistance by showing the potential of deep learning, more particularly on the field of drug discovery.

SNS Telecom & IT estimates that Big Data investments in the healthcare and pharmaceutical industry will account for nearly $4.7 Billion in 2018 alone. Led by a plethora of business opportunities for healthcare providers, insurers, payers, government agencies, pharmaceutical companies and other stakeholders, these investments are further expected to grow at a CAGR of approximately 12% over the next three years.

Furthermore, a significant contribution to disease research roadmap would be support for measures such as tools and databases to integrate and share information, encouraging open access publication and data sharing. For maximum utility and applicability, it is also essential that data is shared so that all information can be used to improve biological modelling. The handling and curating of big data needs improving and depends heavily on these factors. Of course, drug discovery could benefit from this standardization of data across the globe. Many are trying to democratize data, to structure all databases across the global and create a standard registry for drug discovery, in order to keep the homogeneous of the data. With more amount of data becoming accessible, deep learning becomes even more important on this field, due to the fact that it is the approach which deals the best with big data.

Last but not least, as the big data era enters drug discovery research, the development of novel computational concepts for analysis, organization, integration, and utilization of biological and chemical data will be essential. Going forward, cloud computing is expected to play a major role in handling big data. These are challenging and exciting times for computer-aided drug discovery.

Everything indicates that the future of computer-aided drug discovery will be promising. In general, such methods have the greatest potential to be widely applied in the practice of drug discovery, a pre-requisite for success. If we consider that computer-aided drug discovery continues to be driven by experts, as discussed above, a major step forward for this field would indeed be, the generation of chemically intuitive and robust computational methods that become an integral part of day-to-day discovery efforts. The results obtained here prove that our model can be further used to predict whether, the interaction between a new drug and an existing target, or between a new target and some existing drug. Deep learning, like suggested on the course of this work, will have a major role on this revolution. It is necessary to continue exploring this techniques possibilities and how could be applied to drug discovery.

But deep learning reality is already changing. DL was born as the intention to replicate human behaviour on a machine, although unlike these systems, humans learn to actively perceive patterns by sequentially directing attention to relevant parts of the available data. Near future deep NNs will do so, too. When this happens, many doors will open on the drug discovery field, because most of the time is difficult to collect big amount of data, and so, if NN manage to understand patterns with few examples, huge improvements will be achieved.

To sum up, it is a great time for deep learning enthusiasts and the future can only be promising. DL developers have been neglecting the field of drug discovery, despite having no reasons to do so. We will definitely witness a huge improvement on the primitive process of drug discovery on the upcoming years.

Bibliography

[1] C. Walsh, "Antibiotics: Actions, origins, resistance," *Protein Science*, vol. 13, pp. 3059–3060, jan 2009.

[2] European Centre for Disease Prevention and Control, "Annual epidemiological report 2014. Antimicrobial resistance and healthcare-associated infections," *Ecdc*, p. 28, 2014.

[3] I. Roca, M. Akova, F. Baquero, J. Carlet, M. Cavaleri, S. Coenen, J. Cohen, D. Findlay, I. Gyssens, O. E. Heure, G. Kahlmeter, H. Kruse, R. Laxminarayan, E. Liébana, L. López-Cerero, A. MacGowan, M. Martins, J. Rodríguez-Baño, J. M. Rolain, C. Segovia, B. Sigauque, E. Taconelli, E. Wellington, and J. Vila, "The global threat of antimicrobial resistance: Science for intervention," *New Microbes and New Infections*, vol. 6, no. January 2015, pp. 22–29, 2015.

[4] European Centre for Disease Prevention and Control, *The bacterial challenge : time to react*, vol. 6 July 201. 2009.

[5] C. W. Nichole Louise Haag, Kimberly Kay Velk, "Potential Antibacterial Targets in Bacterial Central Metabolism," *Int J Adv Life Sci.*, vol. 6, no. 8, pp. 21–32, 2013.

[6] J. P. Hughes, S. S. Rees, S. B. Kalindjian, and K. L. Philpott, "Principles of early drug discovery," *British Journal of Pharmacology*, vol. 162, pp. 1239–1249, mar 2011.

[7] J. Bajorath, "Computer-aided drug discovery," *F1000Research*, vol. 4, p. 630, 2015.

[8] D. Emig, A. Ivliev, O. Pustovalova, L. Lancashire, S. Bureeva, Y. Nikolsky, and M. Bessarabova, "Drug Target Prediction and Repositioning Using an Integrated Network-Based Approach," *PLoS ONE*, vol. 8, p. e60618, apr 2013.

[9] G. R. Langley, I. M. Adcock, F. Busquet, K. M. Crofton, E. Csernok, C. Giese, T. Heinonen, K. Herrmann, M. Hofmann-Apitius, B. Landesmann, L. J. Marshall, E. McIvor, A. R. Muotri, F. Noor, K. Schutte, T. Seidle, A. van de Stolpe, H. Van Esch, C. Willett, and G. Woszczek, "Towards a 21st-century roadmap for biomedical research and drug discovery: consensus report and recommendations," *Drug Discovery Today*, vol. 22, no. 2, pp. 327–339, 2017.

[10] R. M. Plenge, E. M. Scolnick, and D. Altshuler, "Validating therapeutic targets through human genetics," *Nature Reviews Drug Discovery*, vol. 12, no. 8, pp. 581–594, 2013.

[11] F. K. Brown, F. Kopti, C. Chang, S. A. Johnson, M. Glick, and C. L. Waller, "Data to Decisions: Creating a Culture of Model-Driven Drug Discovery," *The AAPS Journal*, no. 5, 2017.

[12] J. J. Irwin and B. K. Shoichet, "ZINC − A Free Database of Commercially Available Compounds for Virtual Screening ZINC - A Free Database of Commercially Available Compounds for Virtual Screening," *J. Chem. Inf. Model*, vol. 45, no. December 2004, pp. 177–182, 2005.

[13] P. S. Pratik Swarup Das, "A Review on Computer Aided Drug Design in Drug Discovery," *World Journal of Pharmacy and Pharmaceutical Sciences*, no. June, pp. 279–291, 2017.

[14] T. T. Ashburn and K. B. Thor, "Drug repositioning: Identifying and developing new uses for existing drugs," *Nature Reviews Drug Discovery*, vol. 3, no. 8, pp. 673–683, 2004.

[15] N. U. Sahu and P. S. Kharkar, "Computational Drug Repositioning: A Lateral Approach to Traditional Drug Discovery?," *Current topics in medicinal chemistry*, vol. 16, pp. 2069–77, may 2016.

[16] S. Ekins, A. J. Williams, M. D. Krasowski, and J. S. Freundlich, "In silico repositioning of approved drugs for rare and neglected diseases," apr 2011.

[17] J. T. Dudley, T. Deshpande, and A. J. Butte, "Exploiting drug-disease relationships for computational drug repositioning," *Briefings in Bioinformatics*, vol. 12, pp. 303–311, jul 2011.

[18] E. Lounkine, M. J. Keiser, S. Whitebread, D. Mikhailov, J. Hamon, J. L. Jenkins, P. Lavan, E. Weber, A. K. Doak, S. Côté, B. K. Shoichet, and L. Urban, "Large-scale prediction and testing of drug activity on side-effect targets," *Nature*, vol. 486, pp. 361–367, jun 2012.

[19] E. D. Coelho, J. P. Arrais, and J. L. Oliveira, "Computational Discovery of Putative Leads for Drug Repositioning through Drug-Target Interaction Prediction," *PLoS Computational Biology*, vol. 12, no. 11, pp. 1–17, 2016.

[20] M. Wen, Z. Zhang, S. Niu, H. Sha, R. Yang, Y. Yun, and H. Lu, "Deep-Learning-Based Drug-Target Interaction Prediction," *Journal of Proteome Research*, vol. 16, no. 4, pp. 1401–1409, 2017.

[21] F. Yang, J. Xu, and J. Zeng, "Drug-target interaction prediction by integrating chemical, genomic, functional and pharmacological data," *Pac Symp Biocomput*, pp. 148–159, 2014.

[22] F. Cheng, C. Liu, J. Jiang, W. Lu, W. Li, G. Liu, W. Zhou, J. Huang, and Y. Tang, "Prediction of drug-target interactions and drug repositioning via network-based inference," *PLoS Computational Biology*, vol. 8, no. 5, 2012.

[23] Z. He, J. Zhang, X.-H. Shi, L.-L. Hu, X. Kong, Y.-D. Cai, and K.-C. Chou, "Predicting Drug-Target Interaction Networks Based on Functional Groups and Biological Features," *PLoS ONE*, vol. 5, no. 3, p. e9603, 2010.

[24] L. Wang, Z.-H. You, X. Chen, X. Yan, G. Liu, and W. Zhang, "RFDT: A Rotation Forest-based Predictor for Predicting Drug-Target Interactions Using Drug Structure and Protein Sequence Information," *Current Protein & Peptide Science*, vol. 19, pp. 445–454, mar 2018.

[25] A. C. Ian Goodfellow, Yoshua Bengio, *Deep learning*. MIT Press, 2016.

[26] B.J. Copeland, "Artificial intelligence," 2018.

[27] Giancarlo Zaccone, *Getting Started With TensorFlow*. 2017.

[28] A. Gulli and S. Pal, *Learning Deep Learning with Keras*. 2017.

[29] Y. Bengio, "Learning Deep Architectures for AI," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[30] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[31] B. Marr, "What is the difference between artificial intelligence and neural networks?," 2017.

[32] F. Chollet, *Deep Learning with Python*. 2017.

[33] J. Schmidhuber, "Deep Learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances In Neural Information Processing Systems*, pp. 1–9, 2012.

[35] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *Ieee Signal Processing Magazine*, no. November, pp. 82–97, 2012.

[36] P. Mamoshina, A. Vieira, E. Putin, and A. Zhavoronkov, "Applications of Deep Learning in Biomedicine," *Molecular Pharmaceutics*, vol. 13, no. 5, pp. 1445–1454, 2016.

[37] G. Ditzler, R. Polikar, and G. Rosen, "Multi-Layer and Recursive Neural Networks for Metagenomic Classification," *IEEE Transactions on NanoBioscience*, vol. 14, pp. 608–616, sep 2015.

[38] Y. Hadad, "30 amazing applications of deep learning - Yaron Hadad," 2017.

[39] J. C. Pereira, E. R. Caffarena, and C. N. Dos Santos, "Boosting Docking-Based Virtual Screening with Deep Learning," *Journal of Chemical Information and Modeling*, vol. 56, no. 12, pp. 2495–2506, 2016.

[40] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure-activity relationships," *Journal of Chemical Information and Modeling*, vol. 55, no. 2, pp. 263–274, 2015.

[41] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke, "The rise of deep learning in drug discovery," *Drug Discovery Today*, vol. 23, no. 6, pp. 1241–1250, 2018.

[42] A. Koutsoukas, K. J. Monaghan, X. Li, and J. Huan, "Deep-learning: Investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data," *Journal of Cheminformatics*, vol. 9, no. 1, pp. 1–13, 2017.

[43] M. H. Segler and M. P. Waller, "Neural-Symbolic Machine Learning for Retrosynthesis and Reaction Prediction," *Chemistry - A European Journal*, vol. 23, no. 25, pp. 5966–5971, 2017.

[44] D. S. Wishart, C. Knox, A. C. Guo, D. Cheng, S. Shrivastava, D. Tzur, B. Gautam, and M. Hassanali, "DrugBank: A knowledgebase for drugs, drug actions

and drug targets," *Nucleic Acids Research*, vol. 36, no. SUPPL. 1, pp. 901–906, 2008.

[45] Y. Yamanishi, M. Araki, A. Gutteridge, W. Honda, and M. Kanehisa, "Prediction of drug-target interaction networks from the integration of chemical and genomic spaces," *Bioinformatics*, vol. 24, no. 13, pp. 232–240, 2008.

[46] X. Chen, M. Liu, and M. K. Gilson, "BindingDB: a web-accessible molecular recognition database.," *Combinatorial chemistry & high throughput screening*, vol. 4, pp. 719–25, dec 2001.

[47] J. Yang, A. Roy, and Y. Zhang, "BioLiP: a semi-manually curated database for biologically relevant ligand–protein interactions," *Nucleic Acids Research*, vol. 41, pp. D1096–D1103, oct 2012.

[48] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, and G. R. Hutchison, "Open Babel: An open chemical toolbox," *Journal of Cheminformatics*, vol. 3, no. 1, p. 33, 2011.

[49] D. S. Cao, Y. Z. Liang, J. Yan, G. S. Tan, Q. S. Xu, and S. Liu, "PyDPI: Freely available python package for chemoinformatics, bioinformatics, and chemogenomics studies," *Journal of Chemical Information and Modeling*, vol. 53, no. 11, pp. 3086–3096, 2013.

[50] R. Bro and A. K. Smilde, "Principal component analysis," *Anal. Methods*, vol. 6, pp. 2812–2831, apr 2014.

[51] M. Jordan, J. Kleinberg, and B. Schölkopf, "Pattern Recognition and Machine Learning," tech. rep.

[52] E. Gawehn, J. A. Hiss, and G. Schneider, "Deep Learning in Drug Discovery," *Molecular Informatics*, vol. 35, no. 1, pp. 3–14, 2016.

[53] S. Ruder, "An overview of gradient descent optimization algorithms," sep 2016.

[54] A. Moawad, "Neural networks and backpropagation explained in a simple way," 2018.

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.